Why Study TOC :-

→ Mathematical models of a Computer?
  ↳ what cannot a Computer do?
  ↳ How efficiently can a Computer solve?

Turing Machine is a mathematical model of a Computer
⇓
Alan Turing - Father of Computer Science

Turing Prize/award - Award given for Computer Scientists

TOC has tons of realworld applications (TOC, CD)

C-program → $\boxed{\text{GCC}}$ → $\boxed{\text{exe}}$

Eg:- Forms : Email

A-Z @ —— . ——
a-z @ —— . ——
0-9

Error displayed if we Enter invalid patterns

→ Linux/unix : grep → matching patterns

FSA / FSM / Finite state Automata (or) Machine

Automata / Machine / Math-model :-

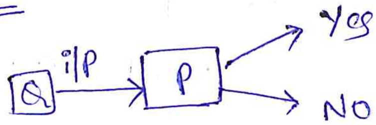
How to denote a pattern : Grammar ← Linguistics

Push-down Automata (PDA) :-

↳ Code → $\boxed{\text{Compiler}}$ → verify Syntax

③ Turing Machines

Halting-problem:



Question is whether Q terminates | halts?

How efficiently can you solve a problem?

(or)

write a program

Deep Turing Machines (DTM)

↑

Building a TM, which can write the programs

Artificial Intelligence

Mathematical preliminaries:-

① **Set :** Collection of distinct objects | elements

↓

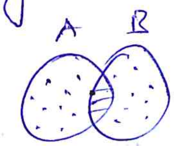
No repetitions $\{1,1,2\}$ ✗

$S = \{1,2,3\}$ ✓

size | cardinality $|S| = 3$ $S' = \{A, B, \ldots Z\}$

$|S'| = 26$

Visually



$A \cup B$

$A \cap B$

Ven-diagram

② **Cartesian product:**

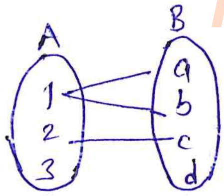
$A = \{1,2\}$ $\Rightarrow$ $A \times B = \{(1,a), (2,a), (1,b), (2,b), (1,c), (2,c)\}$

$B = \{a,b,c\}$

$A \times B = \{(a,b) \mid a \in A, b \in B\}$

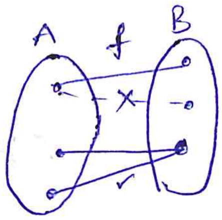
↳ Ordered pairs | Tuples

③ **Relation:** $R \subseteq A \times B$



$R = \{(1,a)(1,b)(2,c)\} \subseteq A \times B$

④ **Function:** It is a special type of relation



$(a,b) \in R$
$(a,c) \in R$ $\Rightarrow b = c$

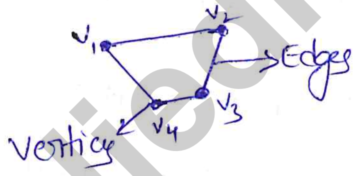
$\forall a \in A$

$\forall b, c \in B$

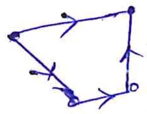
pictorially

⑤ **Graphs:-** It is set of vertices and Edges.

(V, E)



undirected Edges



→ directed Edges

⑥ **Sequences:**

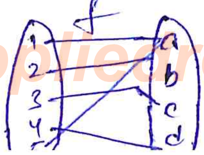
$(A, B, \ldots Z)$: seq of Alphabets
$(1,1,2,3,5 \ldots)$: seq of Fibonacci Number } ordering is important

$\{A, B, C, \ldots Z\}$ ⇒ set of Alphabets
$= \{Z, Y, \ldots D, C, B, A\}$ ordering does not matter

Sequence we can think of a function from natural numbers to objects.

$(1, 1, 2, 3, 5, 8 \ldots)$
$\uparrow\ \uparrow\ \uparrow\ \uparrow\ \uparrow\ \uparrow$
$1\ 2\ 3\ 4\ 5\ 6$ specific ordering

→ Repetitions are allowed in sequences



Formal Languages :- (Linguistics)

Alphabets:

→ Finite Set of Symbols

→ Represented by Σ

English

$\Sigma_{Eng} = \{A, B, \ldots Z, a, b, \ldots z\}$

$\Sigma_{bin} = \{0, 1\}$

$\Sigma_{dec} = \{0, 1, \ldots 9, .\}$

$\Sigma_{Hin} = \{$अ, आ, इ, ...$\}$

$\Sigma_{tel} = \{$అ, ఆ, ఇ, ఈ, ...$\}$

String/word:-

↳ Finite Sequence of alphabets.

abc, 0101, Applied, GATE, 123.45

Length = # alphabets in the String/word.

Eg: $w = abc \Rightarrow |w| = 3$

$w = 0101 \Rightarrow |w| = 4$

Empty String:- $\epsilon, \Lambda, \lambda$

Len = 0 $\Rightarrow |\epsilon| = 0$

$\epsilon \cdot s = s \cdot \epsilon = s$

Σ^* = Set of all the Strings (including Empty Strings) formed over the alphabet Σ

Language: set of strings from the given alphabet.

→ Subset of Σ^*

Grammar:- Set of rules to produce valid Strings in a formal language.

Phone: +91 844-844-0102

Substring:- u is a substring of v if $\exists\ xuy = v$, $u, x, y, v \in \Sigma^*$

Eg: $\underbrace{ab}_{x}\underbrace{cd}_{u}\underbrace{ede}_{y}$

$\underbrace{\epsilon}_{x}\ \underbrace{abcd}_{u}\ \underbrace{ede}_{y}$

prefix:- u is a prefix of v, if $\exists x$ suchthat $ux = v$

$u, x, w \in \Sigma^*$

$\overbrace{\underbrace{01}_{u}0101010 1}^{v}$

concatenation of strings: uv suchthat $u, v \in \Sigma^*$

$u = abc$

$\Rightarrow uv = abcxyz$

$v = xyz$

Finite Automata:

operations on Alphabets, Strings & Languages:

① **power of an Alphabet:-**

let $\Sigma = \{0,1\} \Rightarrow \Sigma^k$

$\uparrow$ Binary Alphabet

$\Sigma^0 = \epsilon$ String of length zero

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

$\Sigma^2 = \Sigma \cdot \Sigma = \{00, 01, 10, 11\}$ Set of all strings of length 2

$\Sigma^3 = \Sigma\Sigma\Sigma = \{000, \ldots\}$ all strings of length 3

$\vdots$

$\Sigma^k = \{w \mid |w| = k\}$ & w are words formed using Σ

② **Kleene Closure:** Σ^*

$\Downarrow$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \ldots$ if $\Sigma = \{0,1\}$

Mail: gatecse@appliedroots.com

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i$

$\Sigma^* = \{ w \mid |w| \geq 0 \}$ w is word formed using Σ

Positive closure:-

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\ldots$, $\Sigma = \{0,1\}$

$\Sigma^+ = \{0, 1, 00, 01, 10, 11, \ldots\}$ does not contain ϵ

$\Sigma^+ = \{ w \mid |w| \geq 1 \}$ w is a word formed using Σ

Basic properties of Σ^+ & Σ^*:

$\Sigma^* = \{\epsilon\} \cup \Sigma^+$

$\Sigma^* \cap \Sigma^+ = \Sigma^+$

$\Sigma^* \cup \Sigma^+ = \Sigma^*$

$\Sigma^* . \Sigma^* = \Sigma^*$

$\Sigma^* . \Sigma^+ = \Sigma^+ = \Sigma^+ . \Sigma^*$

Examples of Languages:-

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

1. $L_1 = \Sigma^*$ = universal Language (Contains all the possible strings in the Language)

2. $L_2 = \Sigma^+ \subseteq \Sigma^*$

3. $L_3 = \{ 0^n \mid n \geq 1 \}$ $\Sigma = \{0\}$ $\subseteq \Sigma^*$

$= \{0, 00, 000, \ldots\ldots\}$

consists of strings of one or more zeros.

④ $L_4 = \{ 0^n 1^n \mid n \geq 1 \} = \{01, 0011, 000111, \ldots\}$

Empty, Finite and Infinite Languages :-

① $L = \{ \} = \phi$ does not even contain ϵ

$|L| = 0$

② **Finite:** $|L|$ is finite $\Rightarrow L = \{ w \mid |w| \leq 2 \}$ over $\Sigma = \{0,1\}$

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

$|L| = 8$

Infinite: $|L|$ is not finite

Finite State Machine/Automata :-

categorization:

- FA
  - FA without outputs
    - DFA → Deterministic
    - NFA (ϵ-NFA)
  - FA with outputs
    - Moore Machine
    - Mealy Machine

What are the applications of FSM/FA (DFA)

ⓐ DFA to determine if a given decimal number is even/odd.

ⓑ Simpler email-address verification FSM

ⓒ Turnstile (wikipedia)

Eg1: Design a FSM to determine if a given decimal number is even/odd

$\Sigma = \{0, 1, \ldots, 9\}$

input $\in \Sigma^*$

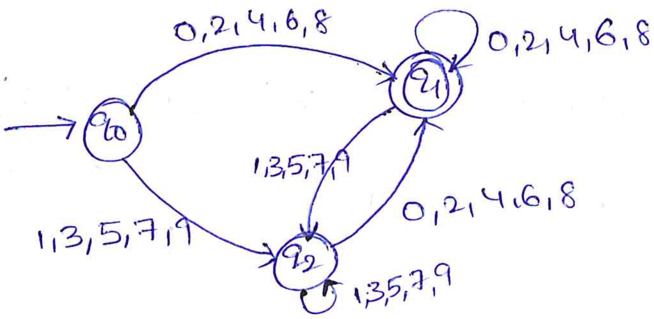
Last digit

let $W_1 = 123$ — 0|2|4|6|8 is Even; 1|3|5|7|9 is odd

$W_2 = 1235670$

input is read from left to right

M:



M Accepts only Even numbers

Transition diagram/ Graph

① $\Sigma = \{0,1,2 \cdots 9\}$

② $Q = \{q_0, q_1, q_2\}$ states

③ Start state q_0

④ Final states : ◎ $\{q_1\}$

zero or more final states

⑤ Transition function

$W_1 = 123$

$\rightarrow q_0 \xrightarrow{1} q_2 \xrightarrow{2} q_1 \xrightarrow{3} q_2$ — M does not accept W_1

$W_2 = 1235670$

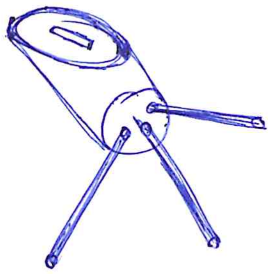
$\rightarrow q_0 \xrightarrow{1} q_2 \xrightarrow{2} q_1 \xrightarrow{3} q_2 \xrightarrow{5} q_2 \xrightarrow{6} q_1 \xrightarrow{7} q_2 \xrightarrow{0} q_1$

final

W_2 is accepted by DFA

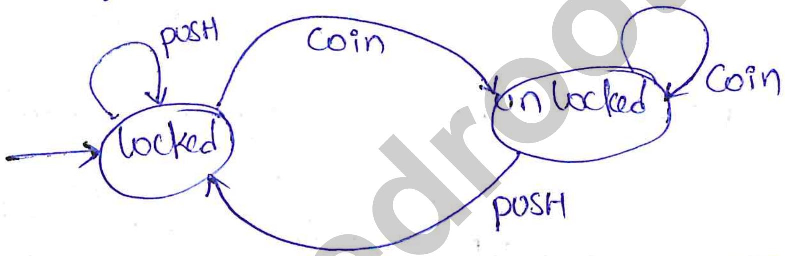
1. .21 invalid
2. 12346 valid - Even
3. 2 valid - Even

Eg 2:- Turnstile machine : we can find in Metro Stations / Airports etc.



After inserting a coin, we are allowed to come out.

State diagram of Turnstile machine



$Q = \{ \text{locked}, \text{un-locked} \}$

$\Sigma = \{ \text{push}, \text{coin} \}$

q_0 (Start state) = locked

$F = \{ \}$

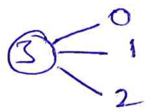
δ : Transition Function

δ	coin	push
locked	unlocked	locked
un-locked	unlocked	locked

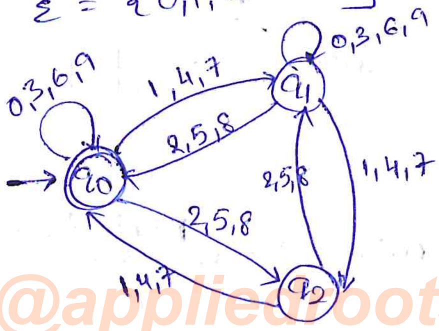
Transition Table

Eg 3: Design a FSM to determine if a decimal number is divisible by 3.

Div by 3



$\Sigma = \{0, 1, \ldots 9\}$



$\Sigma = \{0, 1, \ldots 9\}$

$Q = \{q_0, q_1, q_2\}$

$q_0 = q_0$

$F = \{q_0\}$

δ : Transition Function

lets take an example

① 99

$\rightarrow q_0 \xrightarrow{9} q_0 \xrightarrow{9} q_0$

Final Accepted and divisible by 3.

② 141 $\Rightarrow$ 3|141|47

12
21
21
0

$\rightarrow q_0 \xrightarrow{1} q_1 \xrightarrow{4} q_2 \xrightarrow{1} q_0$

③ 256

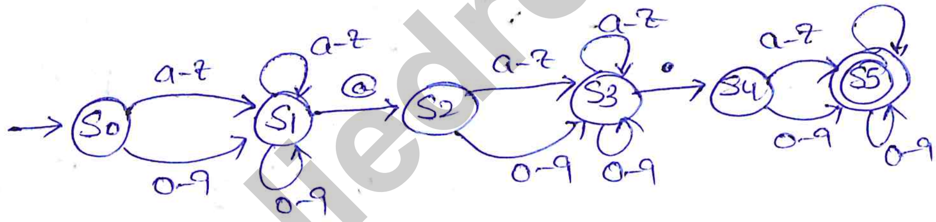
3|256|85

24
16
15
1

$\rightarrow q_0 \xrightarrow{2} q_2 \xrightarrow{5} q_1 \xrightarrow{6} q_1$

Non-Final

Eg 4: DFA to determine a valid email address



[Source: Stackoverflow.com]

* The pattern is

a123 @gmail. com

Representation of a DFA:

① Transition Diagram

② Transition Table $\Rightarrow$

Current State	input: 0	1	2	...	9
q_0					
q_1					
⋮					

Next State

$\rightarrow q_i \xrightarrow{\text{input}} q_j$

* FSM without a Final State
* valid & invalid inputs
* $w_i \in \Sigma^*$

* unique path exist for every input

$$(q_i, x) \rightarrow (q_j)$$

Mathematical Definition of DFA:

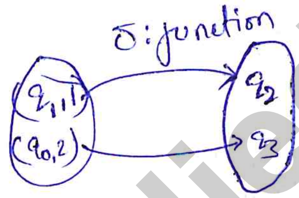
5-tuple | quintuple $\quad M = (\Sigma, Q, q_0, \delta, F)$

- Σ : input alphabet
- Q : set of states (Finite)
- q_0 : initial state
- F : set of final states

δ : Transition Function

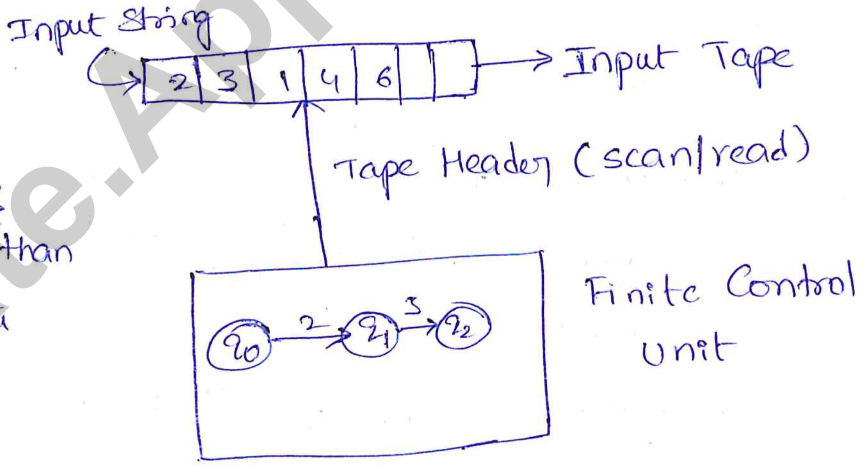
$$\delta : Q \times \Sigma \rightarrow Q$$

$(q_1, 1) \qquad$ Next State



Computation Architecture of FSM:

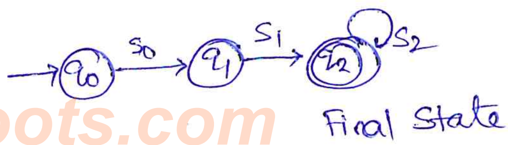
FSM is less powerfull than a computer



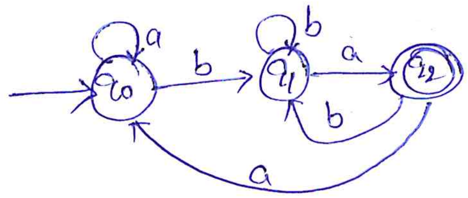
Language of a Finite Automata:

$$L(FA) = \{ w \in \Sigma^* \mid w \text{ is accepted by } FA \}$$

let $w = s_0 s_1 s_2$



Eg 1:- Language of the given Finite Automata over $\Sigma = \{a, b\}$



$W_0 = \epsilon$

$W_1 = a$ ✗ $\quad L = \{ba, aba, aaba, bba, bbba, \dots\}$

$W_2 = b$ ✗ $\quad L = \{W \in \Sigma^* \mid W = xba \text{ where } x \in \Sigma^*\}$

$W_3 = ba$ ✓

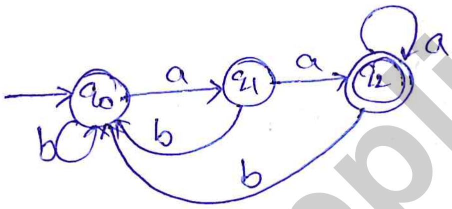
$W_4 = abba$ ✓

$W_5 = aabbbba$ ✓

$W_6 = abbab$ ✗

$W_7 = ababa$ ✓

Eg 2:-



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

$W_1 = ab$ ✗ $\quad q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0$

$W_2 = aa$ ✓ $\quad q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2$ Final

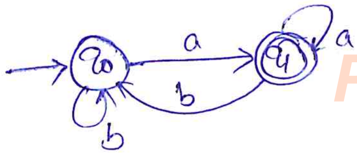
$W_3 = aba$ ✗

$W_4 = abaa$ ✓ $\quad L = \{W \in \Sigma^* \mid W = xaa, x \in \Sigma^*\}$

$W_5 = abb$ ✗ $\quad \Downarrow$

$W_6 = abbaa$ ✓ $\quad L = \{W = xaa \mid x \in \Sigma^*\}$

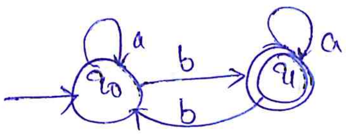
Eg:-3



$L = \{a, aa, \ldots\}$

$= \{xa \mid x \in \Sigma^*\}$

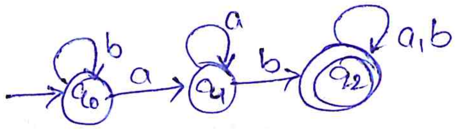
Eg 4



$L = \{b, ba, bbb, \ldots\}$

↳ accept words/strings that have odd number of b's

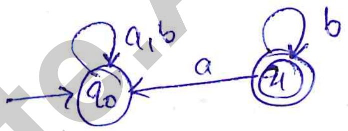
Eg 5:



$L = \{ab, bab, aab, aba, abb \ldots\}$

$L = \{xaby \mid x, y \in \Sigma^*\}$

Unreachable State:

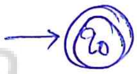


unreachable state

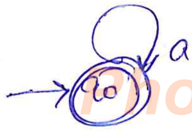
$L = \{\ \} = \phi$

$L = \{\ \} = \phi$

Eg: 1

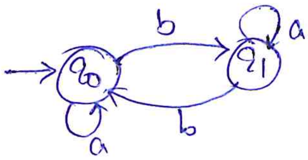


$L = \{\epsilon\}$



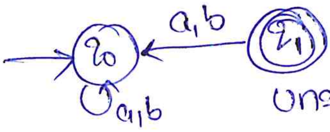
$L = \{\epsilon, a, aa, \cdots\}$

Eg 2:-



No final State $L = \{\ \} = \phi$

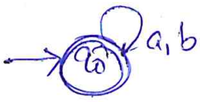
Eg 3:-



Unreachable

$L = \{\ \} = \phi$

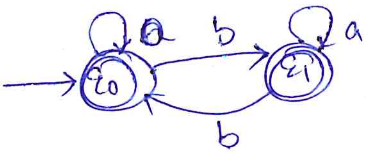
Eg 4:-



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

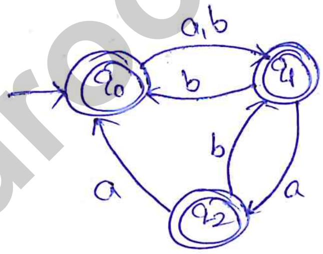
$L = \Sigma^*$

Eg 5:



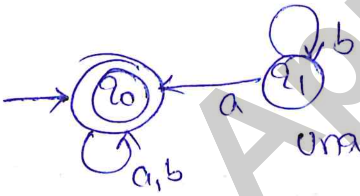
$L = \Sigma^*$

Eg 6:



$L = \Sigma^*$

Eg 7:



Unreachable

→ Every DFA accepts exactly one Language

→ L: Multiple DFA that accepts all words in L

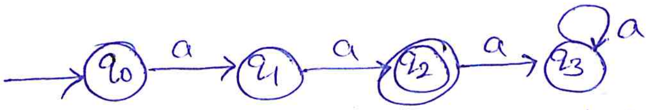
→ Min DFA exactly one

⇓

min # States

Construction of FA for Finite Languages:

Eg 1: $L = \{aa\}$ $\Sigma = \{a\}$



Dead state: No way to come out from the state

ϵ ✗

a ✗

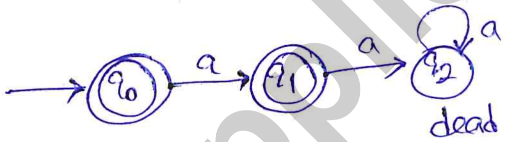
aa ✓

aaa ✗

aaaa ✗

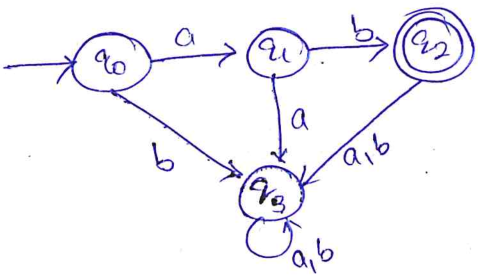
DFA: $\delta : Q \times \Sigma \rightarrow Q$ we need to define the transition for each input symbol at each & every state.

Eg 2: $L = \{\epsilon, a\}$ $\Sigma = \{a\}$



dead state

Eg 3: $L = \{ab\}$ $\Sigma = \{a, b\}$



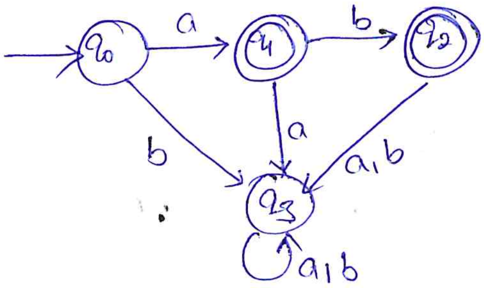
q_3 is a dead state.

DFA will accept only one string "ab"

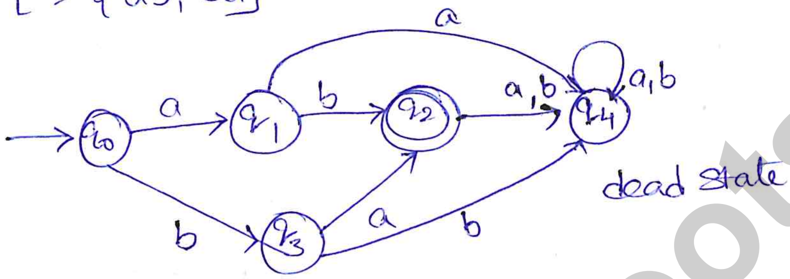
let $w = aba$

$\rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$ invalid / Not accepted String

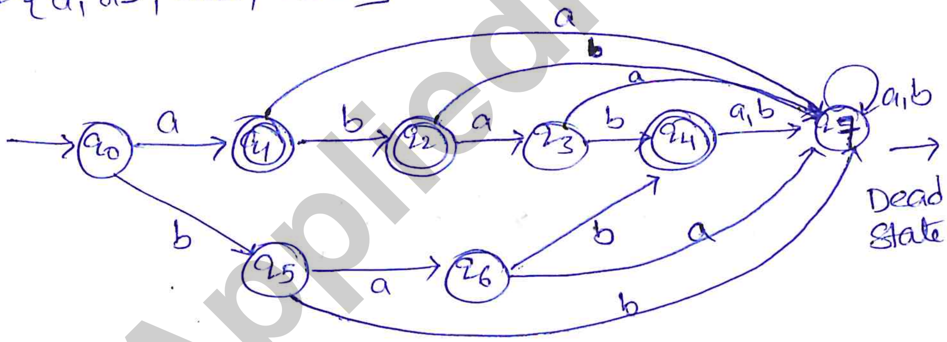
Eg 4: $L = \{a, ab\}$



Eg 5: $L = \{ab, ba\}$



Eg 6: $L = \{a, ab, bab, abab\}$



Regular Languages & Finite Languages:

$L = \{a^n b^n \mid n \geq 1\} \rightarrow$ Non-Regular

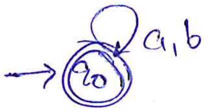
ab, aabb, aaabbb, ...

Regular Language is a Language for which $\exists$ FA that accepts the Language.

All Finite Languages are Regular Language, as we can able to Construct FA for all Finite Language.

Construction of FA for Non-finite Languages: $|L|$ = not finite

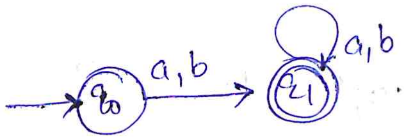
Eg 1: All strings of a's & b's including ϵ over $\Sigma = \{a, b\}$



$L = \Sigma^*$

$L = \{\epsilon, a, b, aa, ab, ba, bb \dots\}$

Eg 2: All strings of a's & b's excluding ϵ $\Sigma = \{a, b\}$

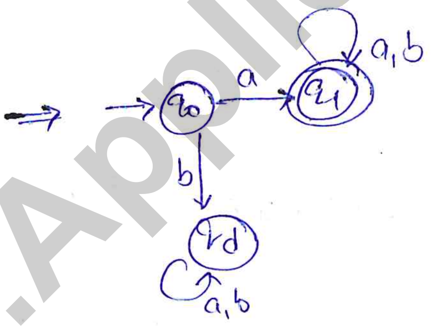


$L = \{a, b, aa, ab, ba, bb, \dots\}$

Eg 3: Strings that start with 'a', $\Sigma = \{a, b\}$

$L = \{ax \mid x \in \Sigma^*\}$

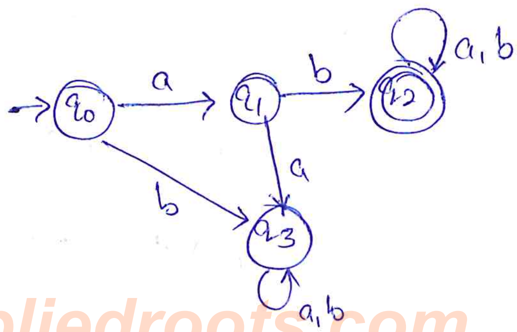
$L = \{a, aa, ab, aaa, abb \dots\}$



Eg 4: Start with 'ab' & $\Sigma = \{a, b\}$

$L = \{abx \mid x \in \Sigma^*\}$

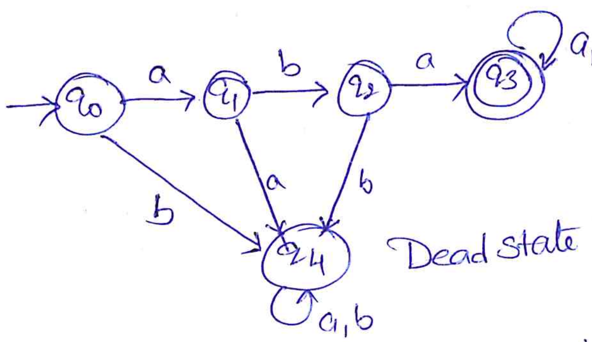
$= \{ab, aba, abb, \dots\}$



<u>Note:</u> In a min FA, there is atmost one dead state

<u>Eg 5:</u> Strings that start with 'aba' & $\Sigma = \{a, b\}$

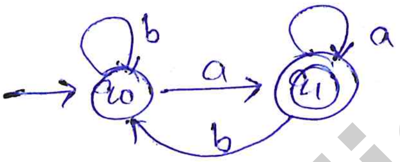
$L = \{abax \mid x \in \Sigma^*\}$



<u>Eg 6:</u> Min FA, that accepts strings ends with 'a'

$\Rightarrow L = \{xa \mid x \in \Sigma^*\}$

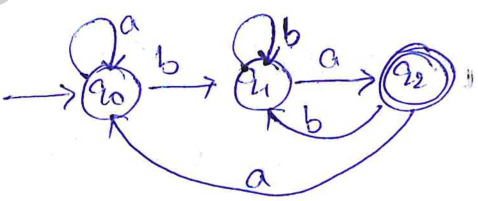
$= \{a, aa, ba, aaa, aba, \ldots\}$



<u>Eg 7:</u> Strings that ends with 'ba'

$\Rightarrow L = \{xba \mid x \in (a,b)^*\}$

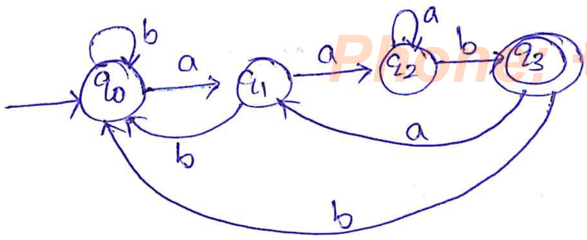
$= \{ba, aba, bba, aaba, abba, \ldots\}$



<u>Eg 8:</u> Strings that ends with 'aab'

$L = \{xaab \mid x \in \Sigma^*\}$

$= \{aab, aaab, baab, \ldots\}$

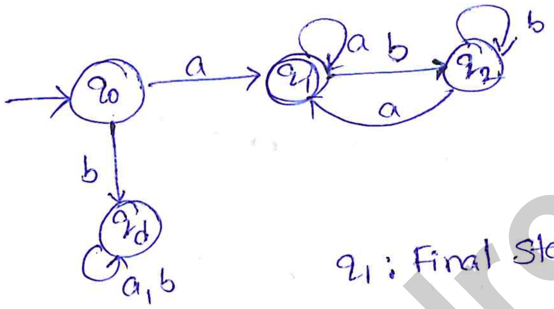


aabbaab

Eg 9: Strings that start & end in 'a' $\Sigma = \{a, b\}$

$$L = \{ axa \mid x \in \Sigma^* \}$$

$$= \{ a, aa, aba, aaa, \ldots \}$$

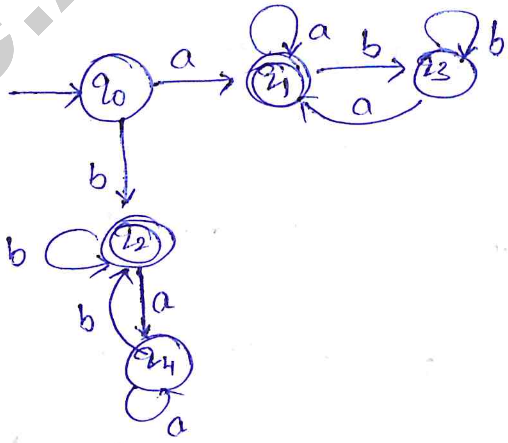


q_1 : Final State

Eg 10:- Strings that start & end with same Symbol

$$L = \{ axa, byb \mid x, y \in \Sigma^* \}$$

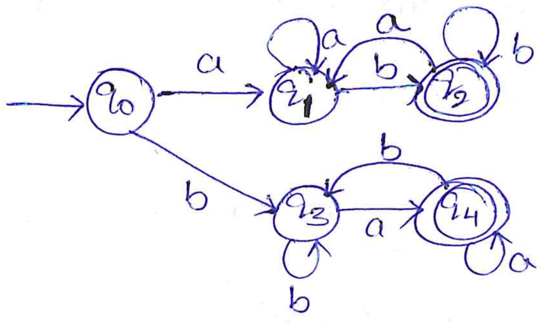
$$= \{ a, b, aaa, aba, bbb, bab, \ldots \}$$



Eg 11: Strings that start & end in different Symbols

$L = \{ axb, bxa \mid x \in \Sigma^* \}$

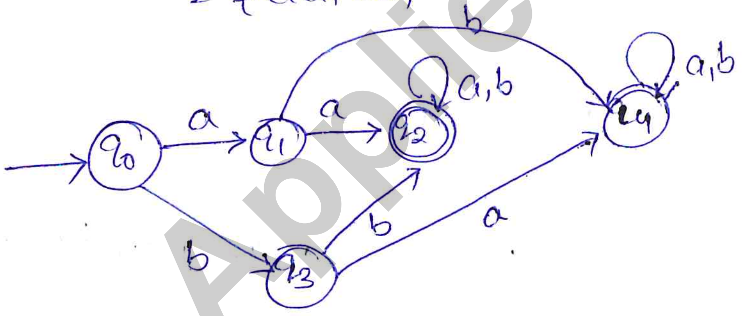
$= \{ ab, ba, aab, abb, baa, bba, \ldots \}$



Eg 12: Strings that start with aa (or) bb

$L = \{ aax, bbx \mid x \in \Sigma^* \}$

$= \{ aa, bb, aab, aaa, bba, bbb, \ldots \}$



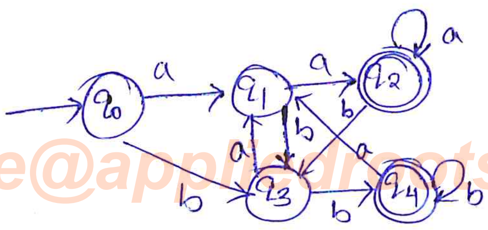
q_2 : Final state

q_4 : Dead State

Eg 13: Strings that end with aa (or) bb

$L = \{ xaa, xbb \mid x \in \Sigma^* \}$

$= \{ aa, bb, aaa, bbb, baa, abb, \ldots \}$



bbaa

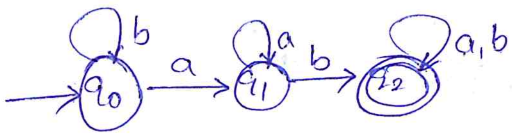
$q_0 \xrightarrow{b} q_3 \xrightarrow{b} q_4 \xrightarrow{a} q_1 \xrightarrow{a} q_2$

F

Eg 14: Strings that contain 'ab' as a substring.

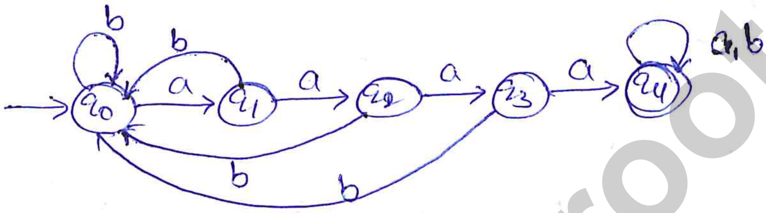
$L = \{ x\,ab\,y \mid x, y \in \{a,b\}^* \}$

$= \{ab, aab, aba, \ldots\}$



Eg 15: Strings that contain 'aaaa' as substring

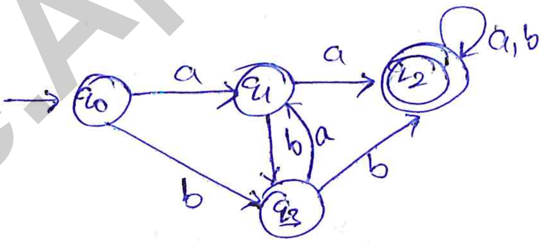
$L = \{ x\,aaaa\,y \mid x, y \in \Sigma^* \}$



Eg 16: Contains aa or bb as a substring

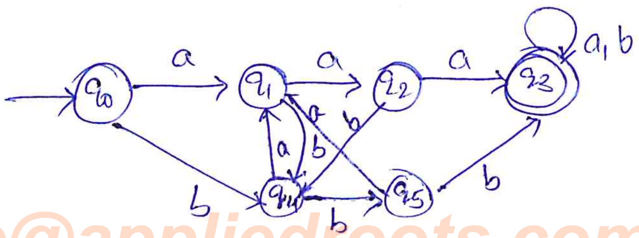
$L = \{ x\,aa\,y, x\,bb\,y \mid x, y \in \Sigma^* \}$

$= \{aa, bb, aaa, bbb, \ldots\}$



Eg 17: Contains aaa (or) bbb as a substring

$L = \{ x\,aaa\,y, x\,bbb\,y \mid x \in \Sigma^* \}$

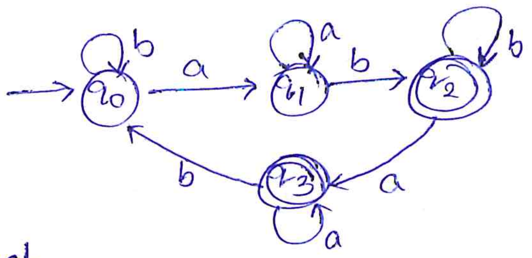


Eg 18: Contains odd occurancy of substring 'ab'

⇓

1, 3, 5, 7 - - - .

L = { ab, aab, aba, ababab, - - - - }



let W = ab

$\rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$

W = aba

$\rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$
F

W = abababa

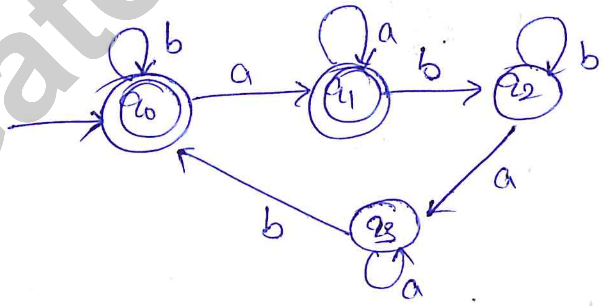
$\rightarrow q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3$
Final

Eg 19: Contains even number of occurancy of substring 'ab'

⇓

0, 2, 4, 6, - - - .

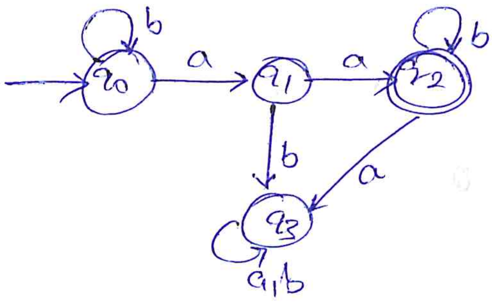
L = { ϵ, abab, abababab, - - - - - }
aaaa, bbb, bbbb



q_0 & q_1 are final states

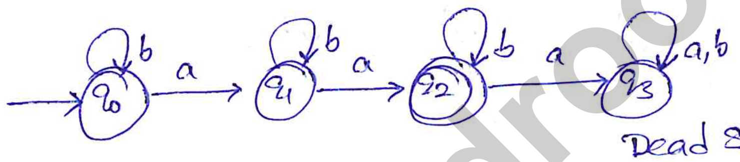
Eg 20: $b^* aa b^*$ all strings containing exactly 2 a's, that are consecutive

L = { aa, baa, aab, ---- }



Eg 21: exactly 2 a's $b^* a b^* a b^*$

L = { aa, baa, aba, ---- }



Eg 22: atmost Two a's

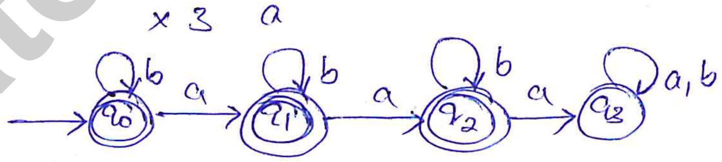
{ ∈, a, aa, ----- }

0 a's

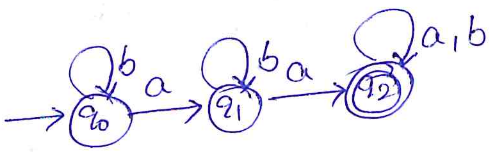
1 a

2 a's

× 3 a



Eg 23:- atleast 2 a's



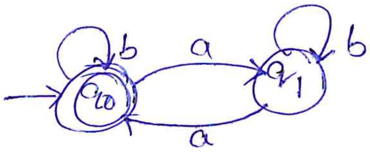
2 a's

3 a's

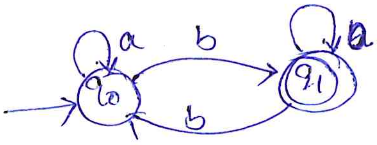
4 a's

⋮

Eg 24:- (a) Even number of a's
⇓
0, 2, 4, 6, - - -



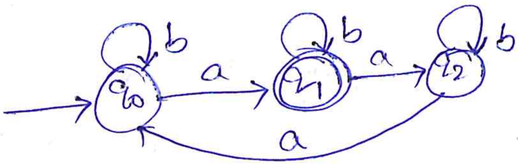
(b) Even number of b's   1, 3, 5, - - -



(c) # a's = 1 mod 3 (Modulo arithmatic)

= 1, 4, 7, 10 - - - -

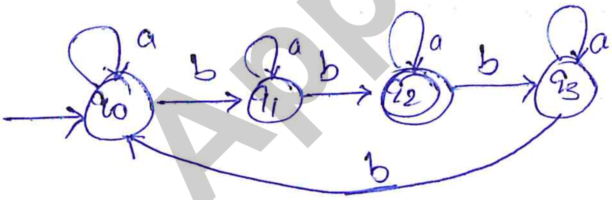
Div 3 → 0, 1, 2   possible reminders



(d) # b's = 2 (mod 4)

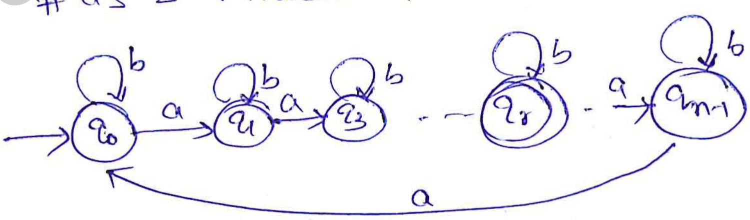
= 2, 6, 10, 14, - - -

Div 4 → 0, 1, 2, 3   possible reminders



Generalization:

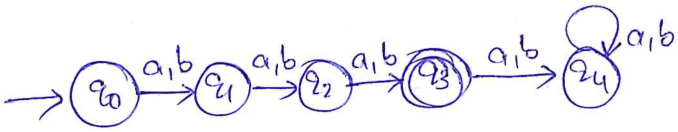
a's = r mod n = |w_a|



n-states   n → 0, 1, 2, ⋮, n-1   possible reminders

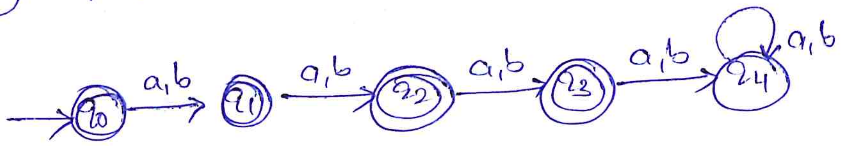
Eg 25: Construct min FA that accepts all strings of $\Sigma = \{a,b\}$ where

(a) $|w| = 3$ (b) $|w| \le 3$ (c) $|w| \ge 3$



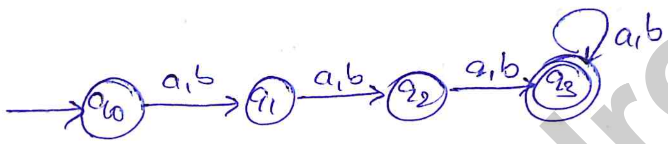
$3+2 \Rightarrow n+2$ States

(b) $|w| \le 3$



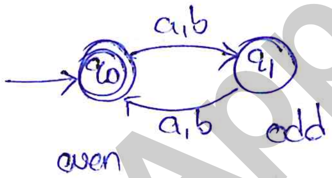
$3+2 \Downarrow n+2$ States

(c) $|w| \ge 3$

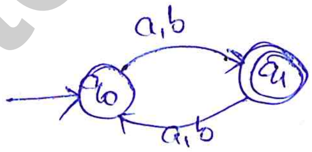


$3+1 \Rightarrow n+1$ States

(d) $|w| = 0 \pmod 2$ 0, 2, 4, ...



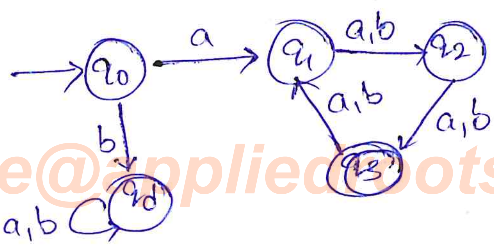
(e) $|w| = 1 \pmod 2$ odd



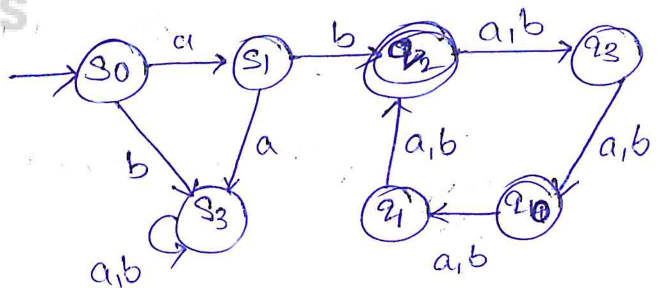
Eg 26: Min FA of a's & b's

(a) Start with 'a' + $|w| = 0 \pmod 3$

$= (0, 3, 6, 9 \cdots)$



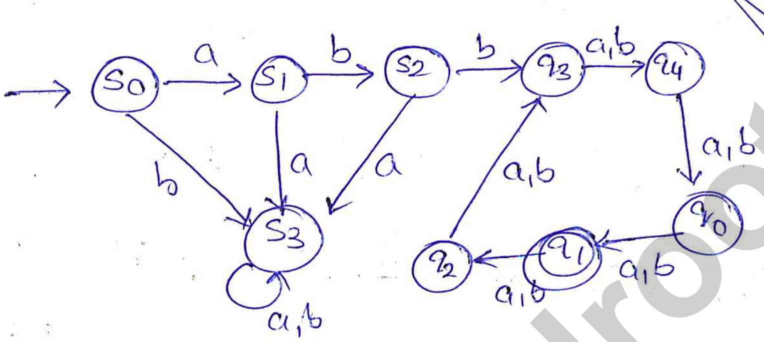
(b) starts with 'ab' & $W = 2 \pmod 4$



4 → 0 – q_0, 1 – q_1, 2 – q_2, 3 – q_3

(c) Starts with 'abb' & $|W| = 1 \pmod 5$

5 → 0, 1, 2, 3, 4

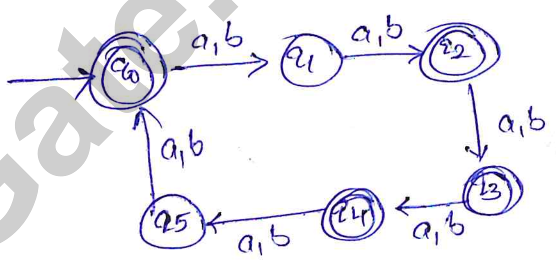


q_1 is Final State

(d) $|W| = 0 \pmod 2$ or $0 \pmod 3$ LCM(2,3) = 6

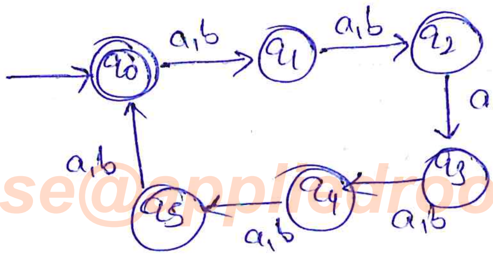
0, 2, 4, 6 ... 0, 3, 6, 9 ...

$\Rightarrow |W| = 0, 2, 3, 4, 6, 8, 9, 10, \ldots$



(e) $|W| = 0 \pmod 2$ and $0 \pmod 3$ LCM(2,3) = 6

6, 12, 18, 24, ...

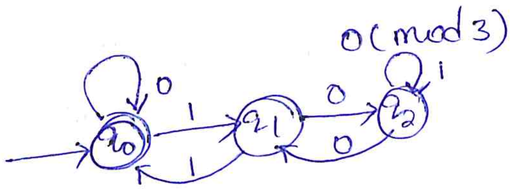


DFA's on binary input: $\Sigma = \{0,1\}$

① Min FA that accepts binary strings whose integer value is $0 \pmod 3$

Binary	Decimal	Rem
✓ 000	0	0
x 001	1	1
x 010	2	2
✓ 011	3	0
x 100	4	1
⋮	⋮	⋮

$0 \pmod 3$



input

States

	0	1
→* q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

4 → 0 — q_0, 1 — q_1, 2 — q_2, 3 — q_3

② Divisible by 4 = $0 \pmod 4$

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

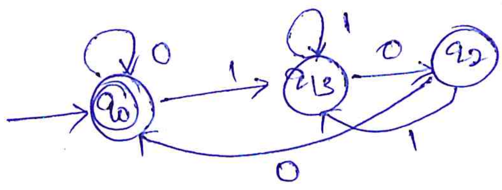
	0	1
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3

Merge (q_1, q_3)

bin	dec	rem
0	0	0
1	1	1
10	2	2
11	3	3
100	4	0
101	5	1
110	6	2
111	7	3

⇓

	0	1
→ q_0	q_0	q_{13}
q_{13}	q_2	q_{13}
q_2	q_0	q_{13}



③ Congruent to 1 mod 6 $\equiv 1 \pmod 6$ → Even

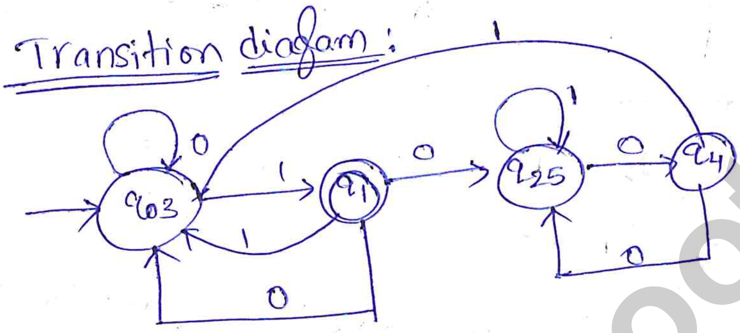
	0	1
→ q_0	q_0	q_1
(q_1)	q_2	q_3
q_2	q_4	q_5
q_3	q_0	q_1
q_4	q_2	q_3
q_5	q_4	q_5

$q_{0,3}$ ⇒

$q_{2,5}$

	0	1
→ q_{03}	q_{03}	q_1
(q_1)	q_{25}	q_3
q_{25}	q_4	q_{25}
q_4	q_{25}	q_{03}

Transition diagram:



$1 \pmod 6 = \{1, 7, 13, \ldots\}$

let $w = 13$
$= 1101$

$\rightarrow q_{03} \xrightarrow{1} q_1 \xrightarrow{1} q_{03} \xrightarrow{0} q_{03} \xrightarrow{1} q_1$
Final state

④ $\equiv 2 \pmod 7$ → odd

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

No two rows are same

⇒ No merging possible

(5) $\equiv 3 \pmod 8$ → Even number 2^3

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

$q_{0,4}$ $q_{1,5}$ $q_{2,6}$

⇒

	0	1
$q_{0,4}$	q_{04}	q_{15}
$q_{1,5}$	q_{26}	q_3
$q_{2,6}$	$q_{0,4}$	q_{15}
(q_3)	$q_{2,6}$	q_7
q_7	q_{26}	q_7

⇒

	0	1
→ q_{0246}	q_{0246}	q_{15}
q_{15}	q_{0246}	q_3
(q_3)	q_{0246}	q_7
q_7	q_{0246}	q_7

Generalization:

Bin number $\equiv r \pmod n$ → minimal FA contains n states

(a) n is odd → minimal FA contains n states

(b) $n = 2^k$ → $k+1$ states

(c) n = even and $n \neq 2^k$, then $n = (2^k) * m$, where m is prime factor of n. then number of states will be $(k+m)$

$\equiv 1 \pmod 6$ ⇒ $(2^1)*3$ ⇒ $k=1, m=3$

number of states $k+m = 1+3 = 4$ states

Trinary Number:

$\Sigma = \{0, 1, 2\}$

① trinary number $\equiv 2 \pmod 4$

	0	1	2
$\rightarrow q_0$	q_0	q_1	q_2
q_1	q_3	q_0	q_1
q_2	q_2	q_3	q_0
q_3	q_1	q_2	q_3

$(210)_3$

$\Rightarrow 3^0 \times 0 + 3^1 \times 1 + 3^2 \times 2$

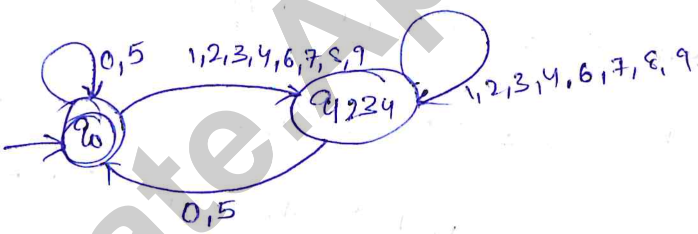
$\Rightarrow 0 + 3 + 18$

$\Rightarrow 21$

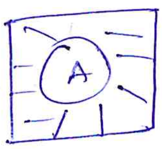
Decimal number (base 10) $\equiv 0 \pmod 5$

	0	1	2	3	4	5	6	7	8	9
$\rightarrow q_0$	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4
q_1	q_0	q_1	q_2	q_3	q_4	q_0	q_1	q_2	q_3	q_4
q_2	⋮	⋮								
q_3	⋮	⋮								
q_4										

Same Transitions

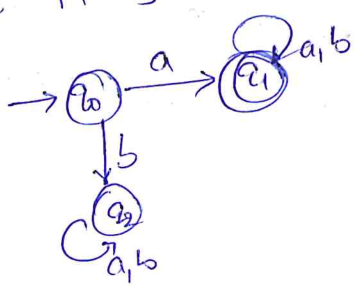


Complement of a FA:

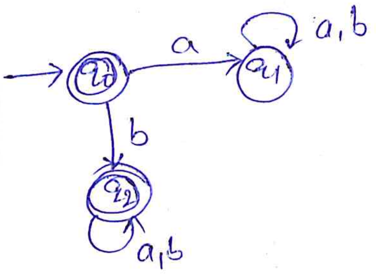


$A^c = A' = U - A$

① let M is



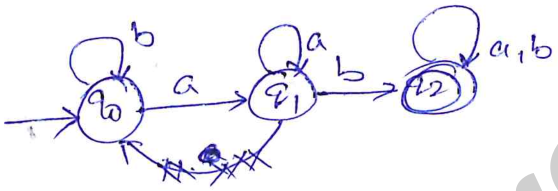
$M' = M^c \Rightarrow L(M') = \Sigma^* - L(M)$



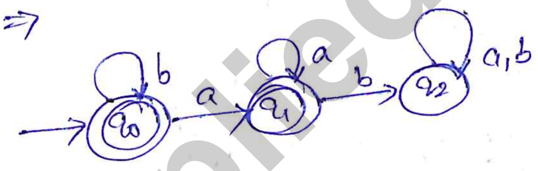
All the Non-final states become the Final states and Final will become Non-Final states.

Eg2: Construct a min FA that accepts all the strings that Donot contain 'ab' as a substring.

$L = \{ w \mid w = xaby \quad x \in \Sigma^*, y \in \Sigma^* \}$



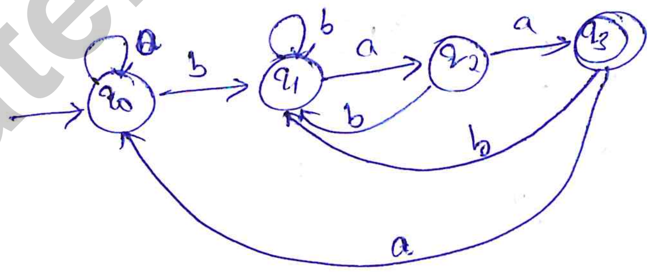
M' ⇒



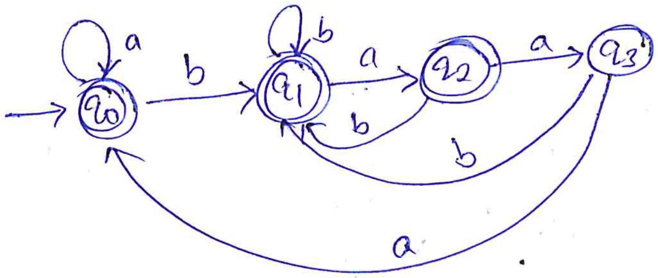
Eg3: Accepts strings that donot end with 'baa'

$L = \{ xbaa \mid x \in \Sigma^* \}$

M



M'



Compound Automata: Unions & Intersections

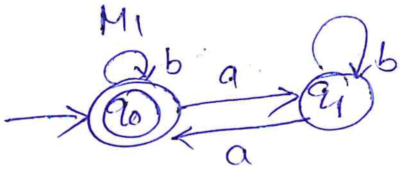
$M_1, L_1 \qquad L_1 \cup L_2 \cup L_3 = L_1 \longrightarrow M_1$

$M_2, L_2 \qquad L_1 \cap L_2 \cap L_3 = L_2 \longrightarrow M_2$

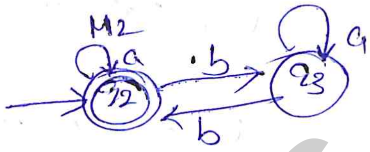
M_3, L_3

(1) $\Sigma = \{a, b\}$ $\underbrace{\text{Number of a's even}}_{L_1}$ $\underbrace{\text{and}}_{\cap}$ $\underbrace{\text{number of b's even}}_{L_2}$

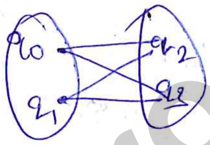
M_1



M_2

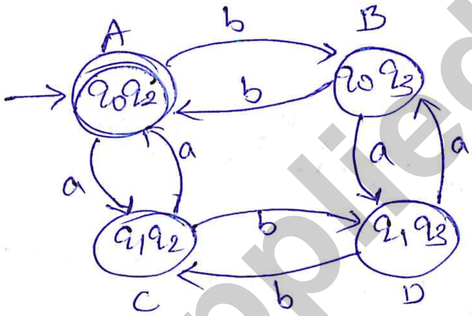


$\Rightarrow L_1 \cap L_2 = L$

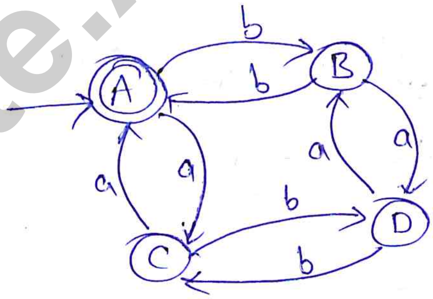


Apply Catesian product

M:



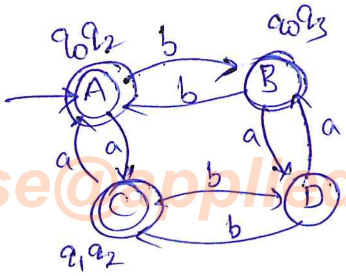
$q_0 q_2$ are final states in M_1, M_2 respectively



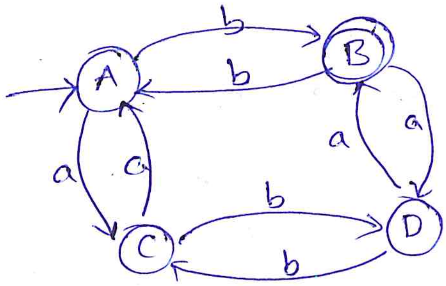
(2) $\underbrace{\#\text{ a's even}}_{L_1}$ $\underbrace{\text{or}}_{\cup}$ $\underbrace{\#\text{ b's even}}_{L_2}$ Final States

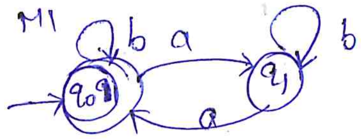
$L = L_1 \cup L_2$

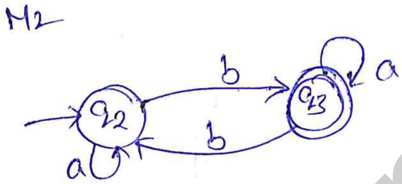
(M)

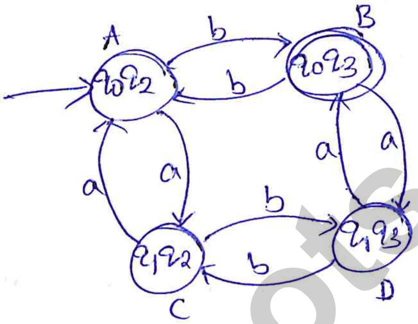


(3) # a's even and # b's is odd.

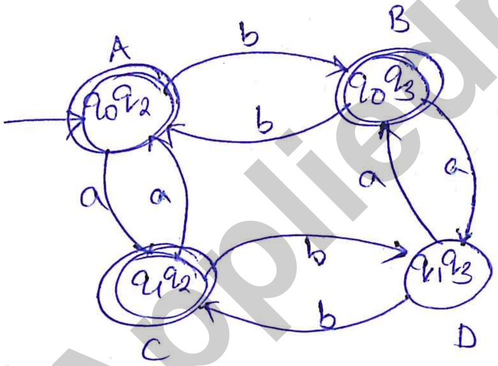






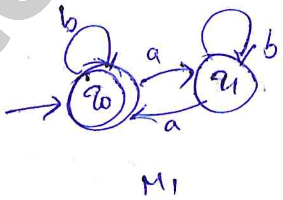


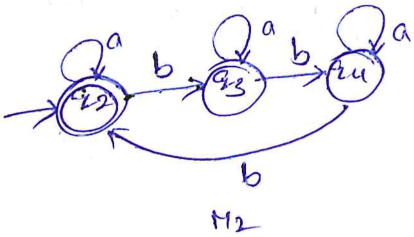
(4) # a's even or # b's is odd

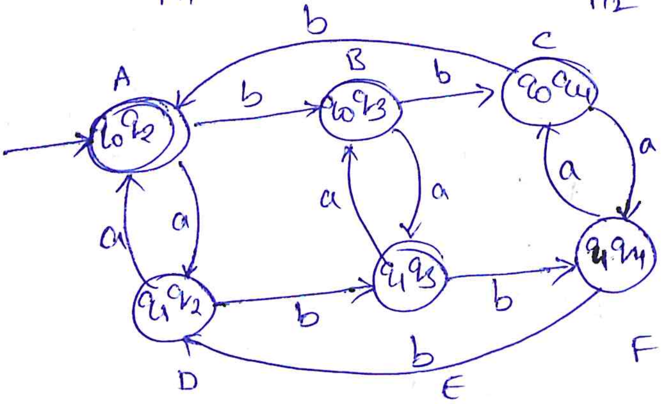


A, B, C are Final State

(5) # a's $\equiv 0 \pmod 2$ and # b's $\equiv 0 \pmod 3$







	a	b
→A	D	B
B	E	C
C	F	A
D	A	E
E	B	F
F	C	D

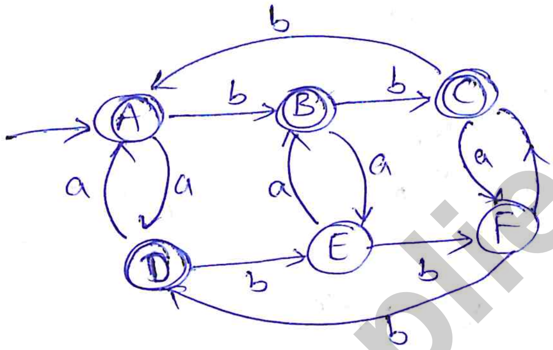
① A {B C D E F}

② C D ✗
F A
A E

③ B C ✗
E C
F A

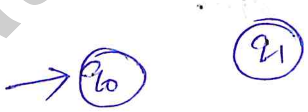
④ E F ✗
B F
C D ✗

⑥ # a's ≡ 0(mod 2) (&) # b's ≡ 0(mod 3)



Counting DFA:-

① # of 2-state DFA with a designated initial state that can be constructed using $\Sigma = \{a, b\}$



① Final States : 2^2

0, 1, 2 ⟶ both $q_0 q_1$ are final states

0 ↓ No Final State

1 ↓ q_0, q_1 may be final

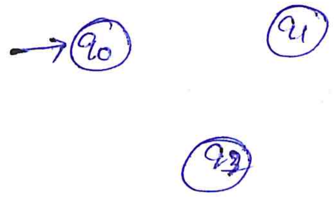
②

δ	a	b
q_0	2	2
q_1	2	2

2^4 possibilities

$2^2 \times 2^4 = 2^6 = 64$ (possible DFA's)

② # of 3-state DFA with a designated initial state that can be constructed using $\Sigma = \{a, b\}$



① Final states: $\{\emptyset, q_0, q_1, q_2, q_0q_1, q_1q_2, q_0q_2, q_0q_1q_2\}$

2^3

②

	a	b
$\rightarrow q_0$	3	3
q_1	3	3
q_2	3	3

$\Rightarrow 3^6$

Total # of possible DFA's are $2^3 \times 3^6 = 8 \times 9^3$
$= 5832$

Generalization:

$|\Sigma| = m$ # states $= |Q| = n$ initial state is designated

① Final states : 2^n

② δ: $m \times n$ cells and each cell can fill with n ways.

$n^{m \times n}$

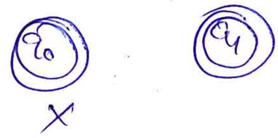
DFA's $\Rightarrow 2^n \times n^{m \times n}$

③ # 2 state DFA's over $\Sigma = \{a, b\}$ that accept $\emptyset$

0 Final state

1 Final state

2 Final States

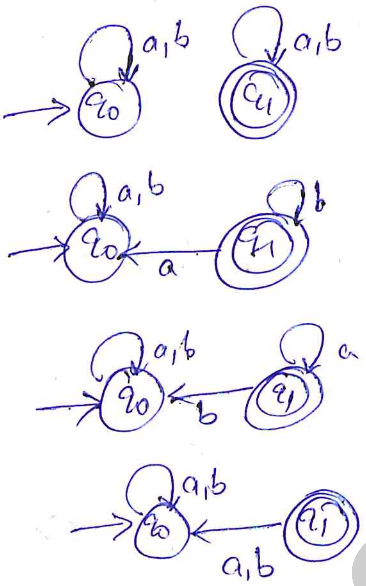


δ:

	a	b
q_0	—	—
q_1	—	—

$2^4 = 16$

16

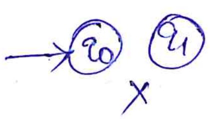


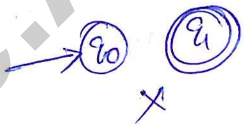
4

⇒ Total # of DFA's possible $= 16 + 4 = 20$

(4) # of 2 state DFAS over $\Sigma = \{a,b\}$ that accept Σ^*

0FS 1F 2F



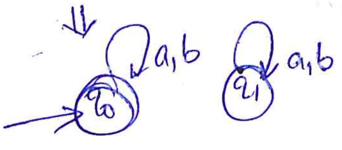


δ	a	b
q_0	2	2
q_1	2	2

$2^4 = 16$ DFA'S

⇓

(1)



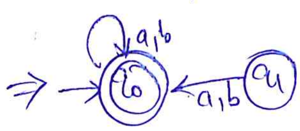
$q_1 \xrightarrow{a,b} q_1$

$q_1 \xrightarrow{a} q_0$

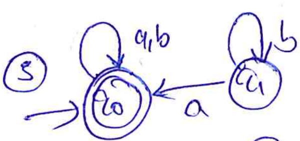
$q_1 \xrightarrow{b} q_0$

Total $= 16 + 4 = 20$

(2)



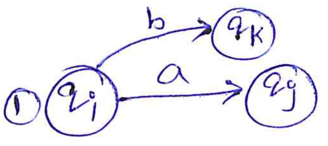
(3)



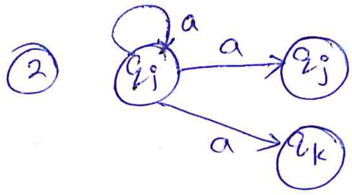
(4)

Non-deterministic FA (NFA)

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



DFA $\quad Q \times \Sigma \rightarrow Q$



NFA

5-Tuple

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

Q : set of states

Σ : input Alphabet

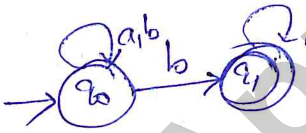
δ : $Q \times \Sigma \rightarrow 2^Q$ (power set of Q)

set of all subsets of Q

$Q = \{q_0, q_1\}$ $\quad |P(Q)| = 2^n$

$P(Q) = 2^Q = \{\phi, q_0, q_1, \{q_0, q_1\}\}$

Eg 1:-



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

$Q = \{q_0, q_1\}$

$P(Q) = \{\phi, q_0, q_1, \{q_0 q_1\}\}$

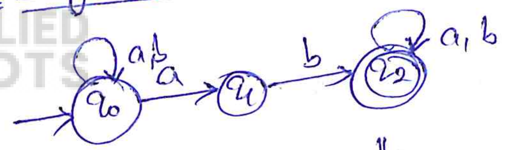
$\delta(q_0, a) = (q_0)$

$\delta(q_0, b) = \{q_0, q_1\}$

$\delta(q_1, a) = \{q_1\}$

$\delta(q_1, b) = \phi$

Acceptance by NFA:

① 

$W_1 = ab$ $\quad q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_0$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$

If any transition path lead to a final state we accept the word.

$W_2 = aa$

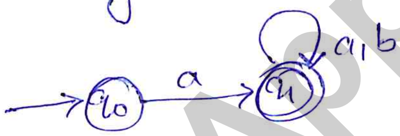
$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0$

$q_0 \xrightarrow{a} q_1$ ✗

Try all possible transition paths

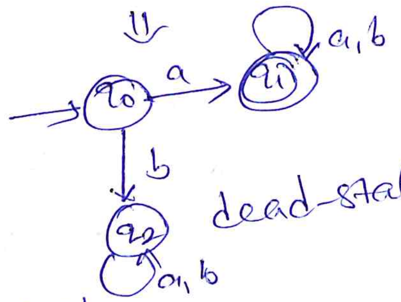
② $\Sigma = \{a, b\}$

ⓐ Strings that start with 'a'



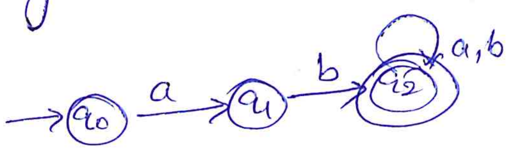
Not DFA $\quad q_0 \xrightarrow{b} \phi$

⇓

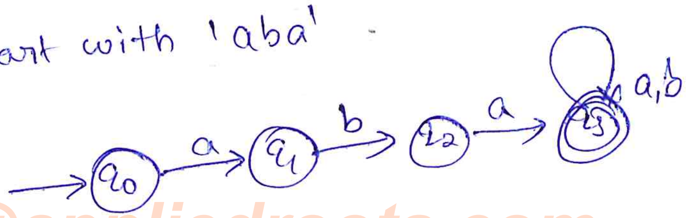


dead-state

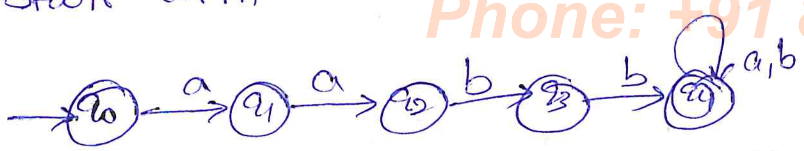
ⓑ Strings that start with 'ab'



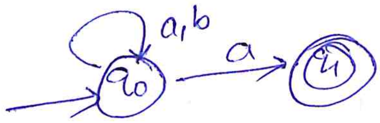
ⓒ Start with 'aba'



(2d) Start with 'aabb'



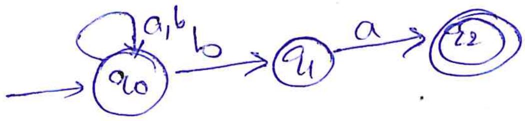
(3a) End with 'a'



$w = ba$

$q_0 \xrightarrow{b} q_0 \xrightarrow{a} \underset{F}{q_1}$ Accepted

(3b) Ends with 'ba'

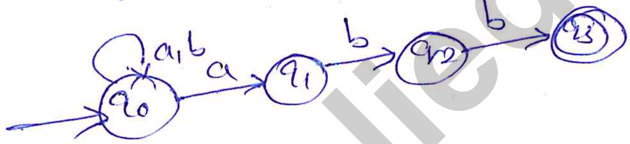


$w = aaba$

$q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2$

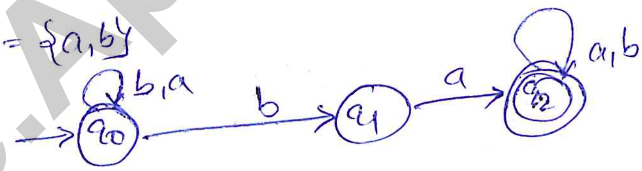
Every NFA $\longrightarrow$ Equivalent DFA

(3c) Ending with abb

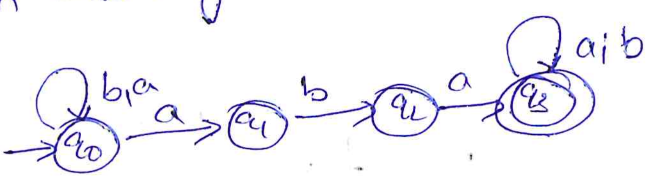


(4a) Contain substring 'ba'

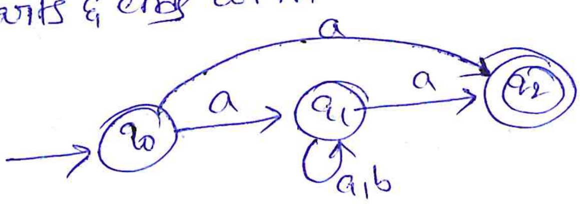
$\Sigma = \{a,b\}$



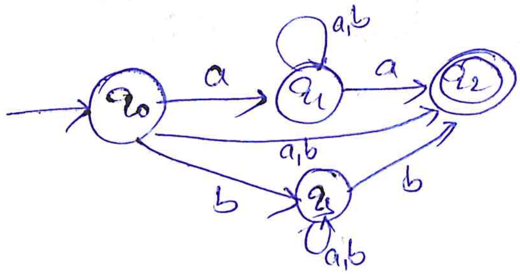
(4b) Contain substring 'aba'



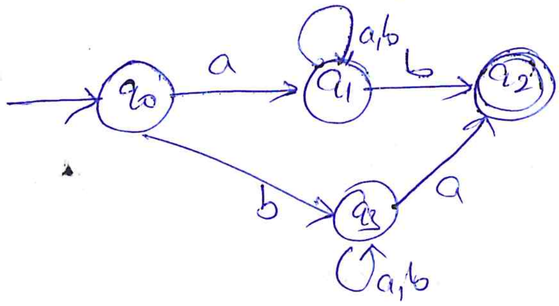
(5a) Starts & ends with a



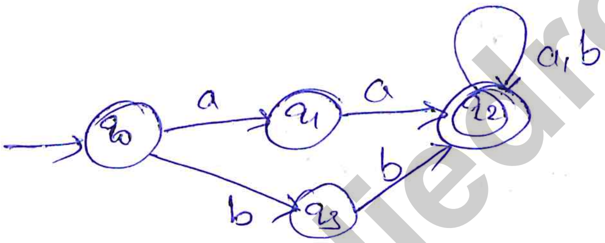
(5b) Starts & ends with Same Symbol



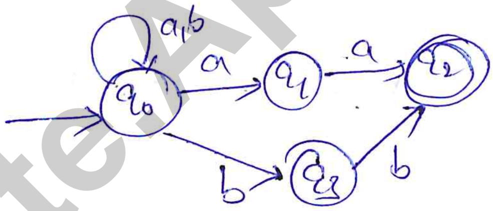
(5c) Starts & ends with different Symbols



(5d) Starts with aa or bb



(5e) Ends with aa or bb



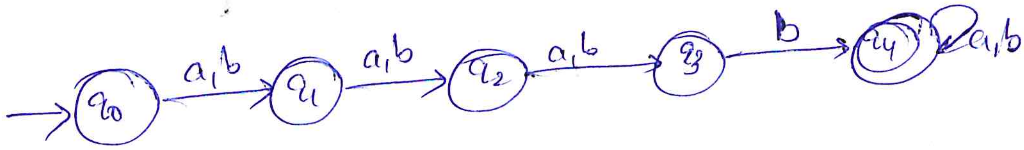
(6a) 4th Symbol from left is 'b'

$w = \underline{x}\ \underline{y}\ \underline{w}\ \underline{b}\ z$
symbols

$W_1 = aaa\underline{b}$ ✓

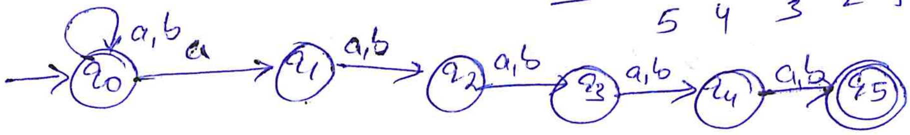
$W_2 = aba\underline{b}abb$ ✓

$W_3 = aaaabb$ ✗

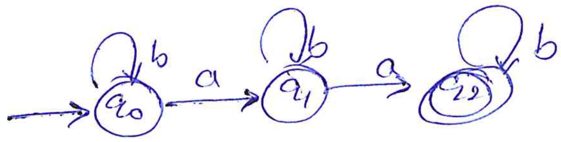


(6b) 5^{th} Symbol from right is 'a'.

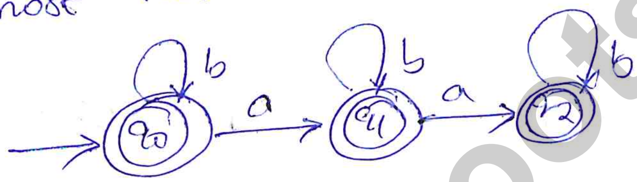
a _ _ _ _
5 4 3 2 1



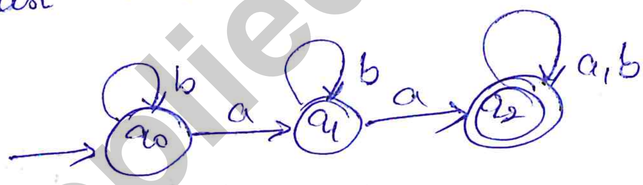
(7a) Contains exactly 2 a's



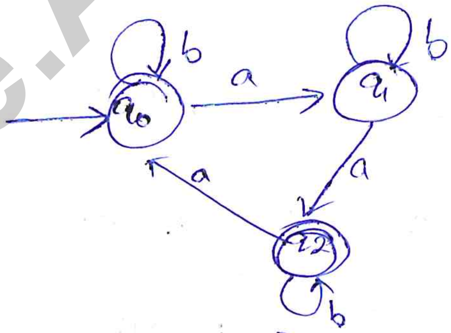
(7b) atmost Two a's



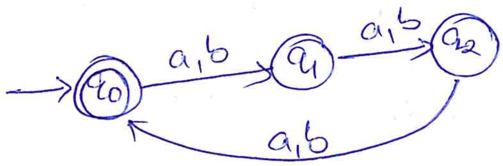
(7c) atleast Two a's



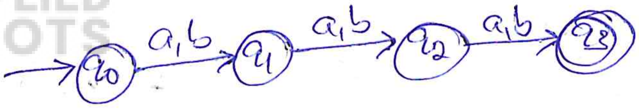
(7d) # a's $\equiv$ 2(mod 3)   NFA & DFA



(8) $|W| \equiv 0 (\bmod 3)$

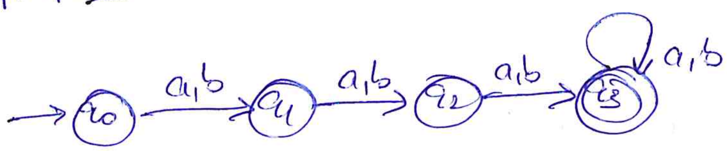


(8a) $|w| = 3$

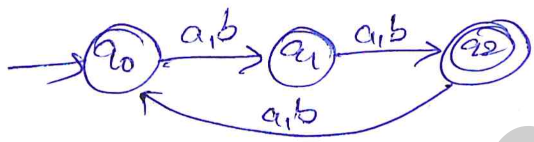


(8b) $|w| \le 3$

(8c) $|w| \ge 3$



(8d) $|w| \cong 2 \pmod 3$. NFA & DFA



Observations:-

→ comp: S/W → DFA is easier to implement & efficient

→ NFA : Equivalent DFA

→ Design of NFA's easier, especially for complex systems

→ NFA : parallel computing engine { multi-threaded }

→ No need of dead states in NFA

→ Multiple transition paths & some transitions are not defined in NFA's.

Conversion from NFA to DFA:

Complex Sys → NFA → DFA → S/W

① NFA

δ	a	b
$\rightarrow q_0$	q_0	q_1
q_1	q_0, q_1	q_2
q_2	q_1	q_0, q_1

⇒

DFA

δ'	a	b
$\rightarrow q_0$	q_0	q_1
q_1	q_0q_1	q_2
q_0q_1	q_0q_1	q_1q_2
q_2	q_1	q_0q_1
q_1q_2	q_0q_1	$q_0q_1q_2$
$q_0q_1q_2$	q_0q_1	$q_0q_1q_2$
q_0q_2	q_0q_1	q_0q_1

need not be the min DFA

$Q = \{q_0, q_1, q_2\}$

2^Q

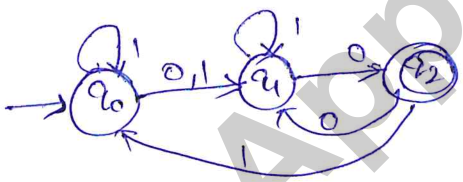
① input symbols

② States

③ Final State

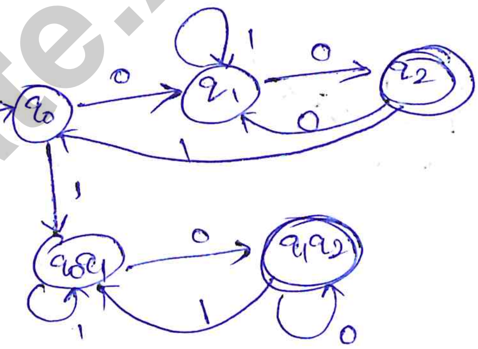
④ Initial State

⑤ δ'

② 

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

Atmost 2^3 States in DFA

DFA 

	0	1
q_0	q_1	q_0q_1
q_1	q_2	q_1

$q_1 \xrightarrow{0} q_2$

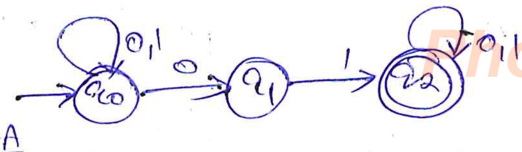
$q_2 \xrightarrow{0} q_1$

$q_1 \xrightarrow{1} q_1$

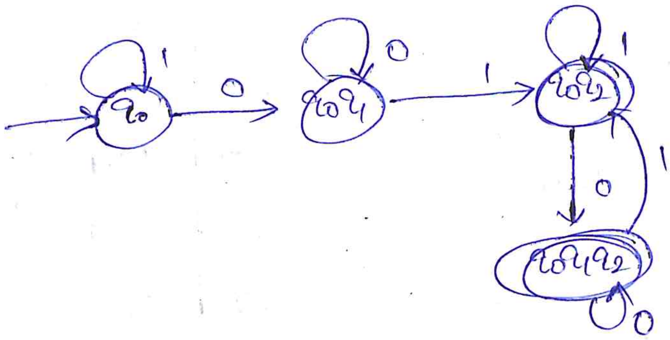
$q_2 \xrightarrow{1} q_0$

Eg 3:

NFA



DFA:



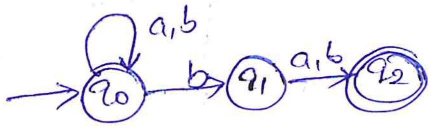
$q_0 \xrightarrow{0} q_0 q_1$

$q_1 \xrightarrow{0} \phi$

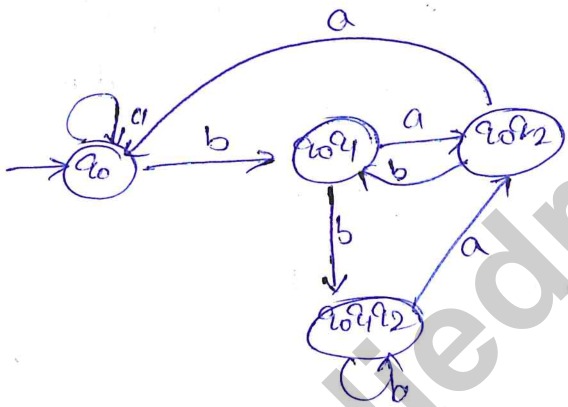
$q_0 \xrightarrow{1} q_0$

$q_1 \xrightarrow{1} q_2$

Eg 4:



DFA:



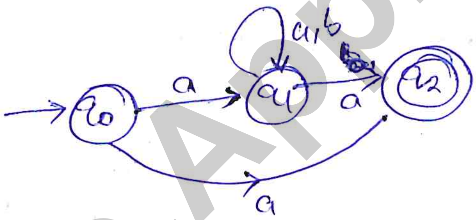
$q_0 \xrightarrow{a} q_0$

$q_1 \xrightarrow{a} q_2$

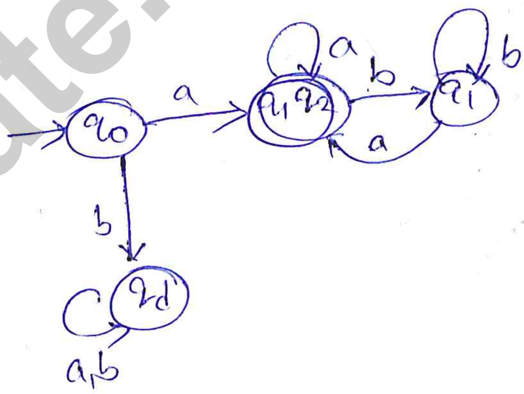
$q_0 \xrightarrow{b} q_0 q_1$

$q_1 \xrightarrow{b} q_2$

Eg 5:



DFA:



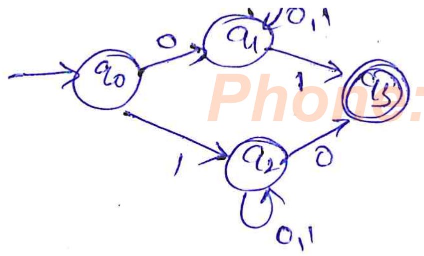
$q_1 \xrightarrow{a} q_1, q_2$

$q_2 \xrightarrow{a} \phi$

$q_1 \xrightarrow{b} q_1$

$q_2 \xrightarrow{b} \phi$

Eg 6: M



① $L = L(M)$

② change Final to Non-final vice-versa $\rightarrow M' \& L'$

③ which options are correct

ⓐ $L = L'$

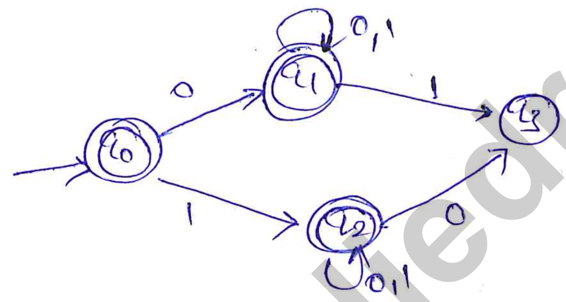
ⓑ $L \subset L'$ ✓

ⓒ $L' = \Sigma^*$ ✓

ⓓ None

$L = 0 \_\_ 1$

$1 \_\_ 0$

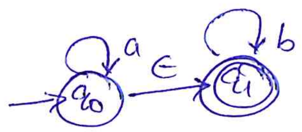


$L' = \{ \epsilon, 0, 1, \ldots \}$

Epsilon-NFA + Conversion from ε-NFA to NFA & DFA

ε-NFA:

Eg1:



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

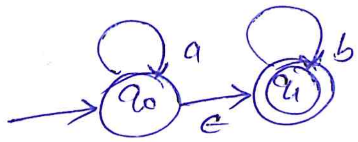
ε transition: By reading Empty string we can move from one state to another state.

NFA: easier to model

ε-NFA: more easier to model

$a \cdot \epsilon = a = \epsilon \cdot a$

$w \cdot \epsilon = w = \epsilon w$



$L = \{ \epsilon, a, aa, \ldots$
$b, b, \ldots$
$ab, aab, aabb \ldots$
$abb \ldots \}$

$L = \{ a^m b^n \mid m, n \geq 0 \}$

power: ϵ-NFA $\sim$ NFA $\sim$ DFA

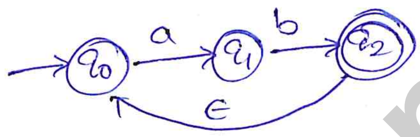
$M = (Q, \Sigma, \delta, q_0, F)$

$\delta : Q \times \Sigma \rightarrow 2^Q$ NFA

$Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$ set of states

State Symbol Empty String

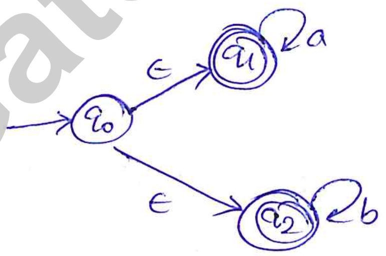
Eg 2:



$L = \{ ab, abab, \ldots \}$

$= \{ (ab)^n \mid n \geq 1 \}$

Eg 3:

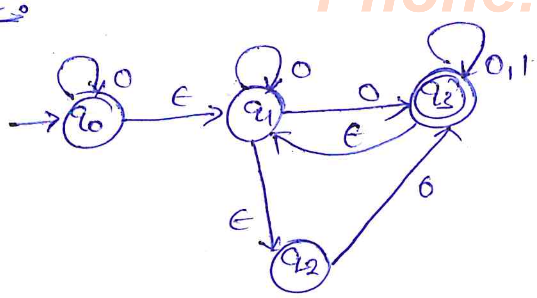


$L = \{ \epsilon, a, aa, aaa \ldots$
$\epsilon, b, bb, bbb \ldots \}$

$= \{ a^n \cup b^n \mid n \geq 0 \}$ (or)

$\{ a^n \mid n \geq 0 \} \cup \{ b^n \mid n \geq 0 \}$

ϵ-closure:



$\delta(q_i, \epsilon) = (q_j)$

ϵ-closure$(q_0) = \{q_0, q_1, q_2\}$

set of all states that can be reached from the state q_i by reading an empty string ϵ is known as ϵ-closure.

ϵ-closure$(q_1) = \{q_1, q_2\}$

ϵ-closure$(q_2) = \{q_2\}$

ϵ-closure$(q_3) = \{q_3, q_1, q_2\}$

Note: ϵ-closure is never an empty set

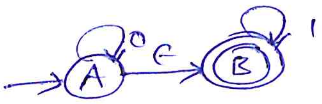
ϵ-closure$(\phi) = \phi$

ϵ-NFA $\xrightarrow{1}$ NFA $\longrightarrow$ DFA

ϵ-NFA $\xrightarrow{2}$ DFA

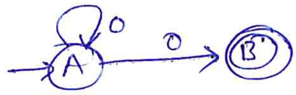
1 ϵ-NFA to NFA Conversion:

Eg1:



$\epsilon(A) = \{A, B\}$

$\Rightarrow$



$\epsilon(B) = \{B\}$

$A' \xrightarrow{0} A \xrightarrow{\epsilon} B$

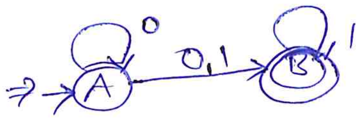
$\delta'(A, 0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A), 0))$

$= \epsilon\text{-closure}(\delta(A,B),0)$

$= \epsilon\text{-closure}(A,\phi)$

$= \epsilon\text{-closure}(A)$

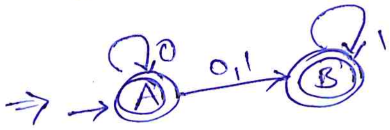
$= \{A,B\}$



$A \xrightarrow{\epsilon} B \xrightarrow{1} B$

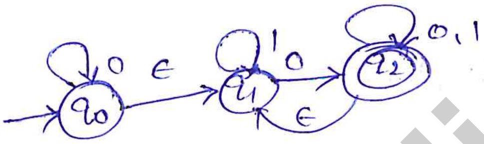
$A \xrightarrow{\epsilon 1} B$

$\Rightarrow A \xrightarrow{1} B$



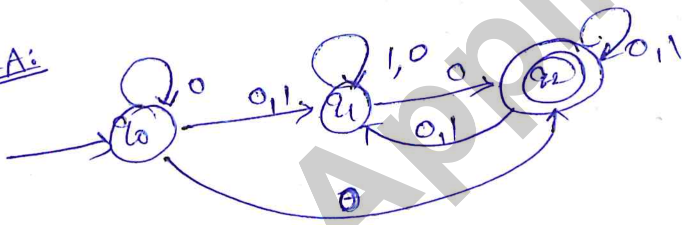
Final State

Eg 2:



ϵ-NFA

NFA:



No change in set of states ✓

initial state ✓

Final state ✓

$\rightarrow q_0 \xrightarrow{0} q_0 \xrightarrow{\epsilon} q_1$

$\rightarrow q_0 \xrightarrow{0} q_0 \xrightarrow{0,1} q_1$

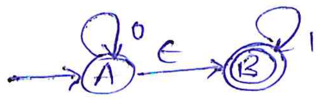
$\rightarrow q_0 \xrightarrow{0} q_2$

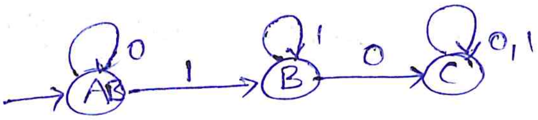
ϵ-NFA to DFA Conversion:

ϵ-NFA $\longrightarrow$ NFA $\longrightarrow$ DFA

Thomson Construction   Subset Construction

Eg1:





Compute ϵ-closure of initial state

$\epsilon(A) = \{A, B\}$

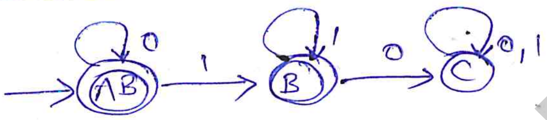
$\Sigma = \{0, 1\}$

$A \xrightarrow{0} A \xrightarrow{\epsilon} B$

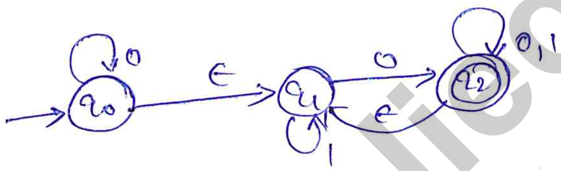
$A \xrightarrow{\epsilon} B \xrightarrow{1} B$

$B \xrightarrow{0}$ No transition

Final State:

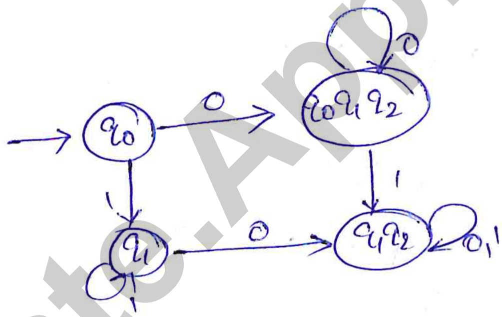


Eg2:



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

ϵ-closure $(q_0) = \{q_0 q_1\}$



$q_0 \xrightarrow{0} q_0 \xrightarrow{\epsilon} q_1$

$q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{0} q_2$

$q_0 \xrightarrow{\epsilon} q_1 \xrightarrow{1} q_1$

$q_0 q_1 q_2 \xrightarrow{0} q_0 q_1 q_2$

$q_0 q_1 q_2 \xrightarrow{1} q_1 q_2$

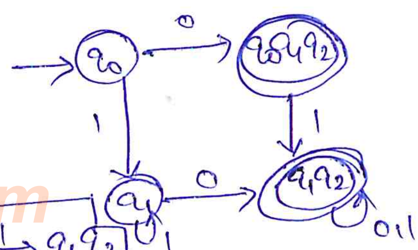
$q_1 \xrightarrow{0} q_2 \xrightarrow{\epsilon} q_1$

$q_1 q_2 \xrightarrow{0} q_1 q_2$

$q_1 q_2 \xrightarrow{1} q_1 q_2$

Final State: q_2

$\left.\begin{array}{l} q_0 q_1 q_2 \\ q_1 q_2 \end{array}\right\}$ Final States in DFA



Properties and Equality of FA:

Decision properties of FA:

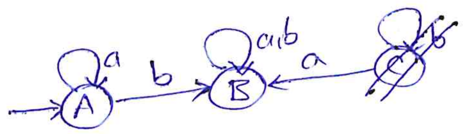
① **Emptiness:-** Does the FA accept empty or non-empty L.

ⓐ Delete/Ignore all unreachable states

ⓑ atleast one final state ⇒ Non-empty L

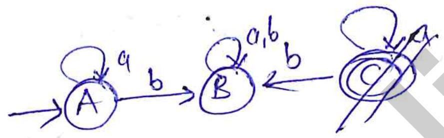
No Final states ⇒ Empty Language

M1 :



unreachable states

M2 :



② **Finiteness:** Does the FA accept finite Language?

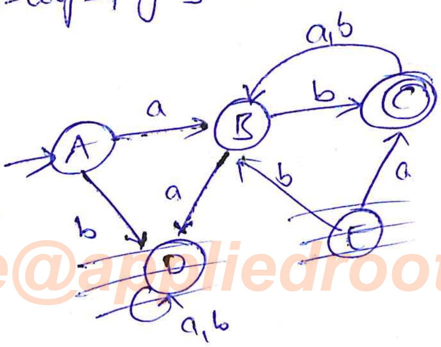
ⓐ Ignore/delete unreachable states

ⓑ Ignore/delete dead-states & all other states from which we cannot reach a Final State

ⓒ Loops/Cycles ⇒ infinite-Language

ⓓ No-loops/cycles ⇒ Finite Language.

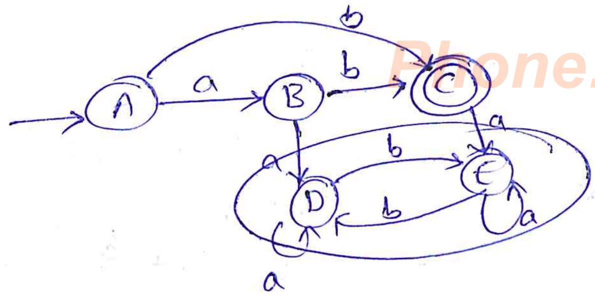
Eg:



M1 : Infinite Language.

ignore unreachable & dead states

M_2



Ignore Dead States

The Language accepted by FA is $\{b, ab\}$

Finite Language.

③ <u>Membership / Acceptance :</u>

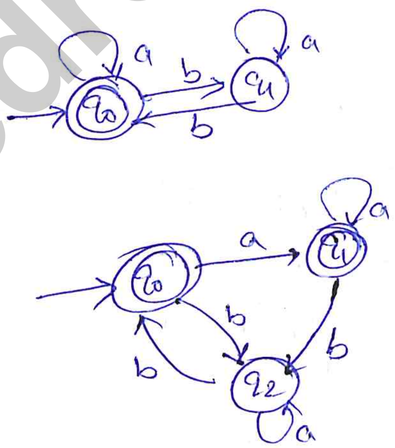
$q_0 \to q_1$

$\rightsquigarrow$ Transition path

(W) $\boxed{M}$

if $\exists$ $q_0 \overset{W}{\rightsquigarrow}$ final state then w is accepted by M

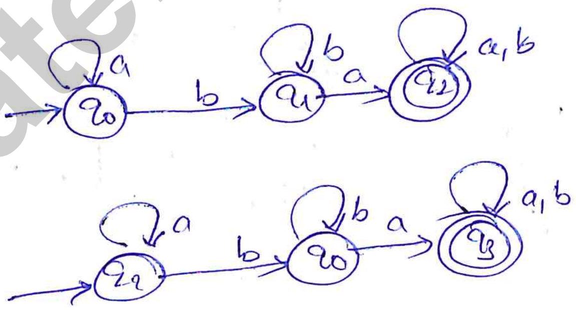
④ <u>Equality of M_1 & M_2 :</u>



$M_1 = M_2$ iff

$L(M_1) = L(M_2)$

<u>Isomorphism :</u>



M_1

M_2

are Isomorphic to each other.

$|Q_1| = |Q_2|$

$L(M_1) = L(M_2)$

{ By changing the names / lables of the states, we can able get the second machine. }

Quintuple δ_1 $Q_1 = \{q_0, q_1, q_2\}$

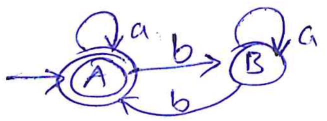
$\#$ $\qquad \#$

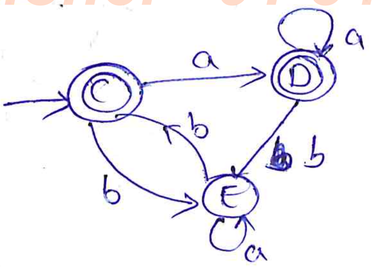
δ_2 $Q_2 = \{q_2, q_0, q_3\}$

52

Test for equality of two machines:

Eg 1:





		a	b
F,F	A,C	AD	BE
F,F	A,D	AD	BE
NF,NF	B,E	BE	AC

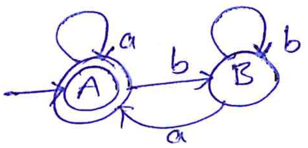
FF ✓

NF NF ✓

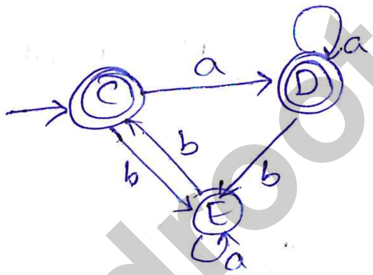
F NF ✗

NF F ✗

Eg 2:



$\neq$

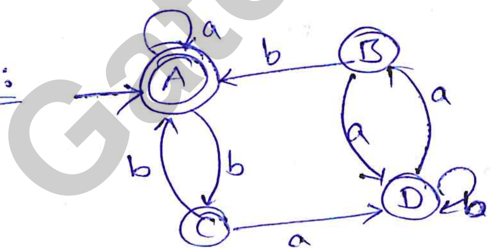


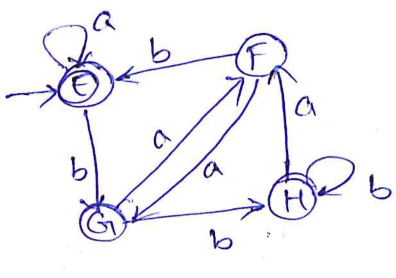
	a	b
AC	AD	BE
AD	AD	BE
BE	AE	BC

F NF

✗

Eg 3:





	a	b
AE	AE	CG
CG	DF	AH

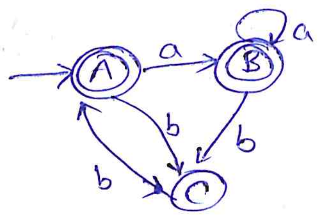
AH → NF

AH → F

$M_1 \neq M_2$

DFA Minimization: (Wiki) → Multiple-algorithm → pseudo-code

Eg1:



	a	b
→ A (final)	B	C
B (final)	B	C
C	–	A

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

① Delete all unreachable states

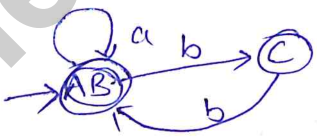
② Merge final states & NF-states

	a	b
A (final)	B	C
B (final)	B	C

⇒ A & B are Final states and equal states

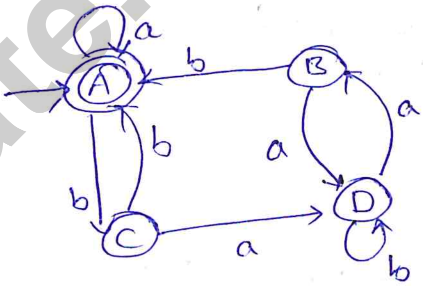
	a	b
→ AB (final)	AB	C
C	–	AB

⇒



AB is the Final State

Eg2:



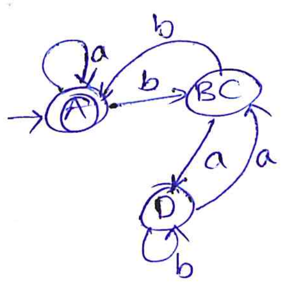
⇒

	a	b
→ A (final)	A	C
B	D	A
C	D	A
D	B	D

	a	b
→ A (final)	A	BC
BC	D	A
D	BC	D

→ F
→ NF

⇒

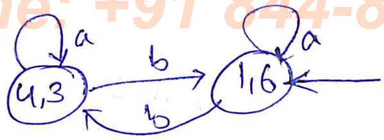


A is Final State

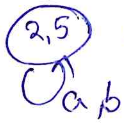
Eg 3:

M1

δ	a	b
→1	6	4
2	5	2
3	$\boxed{4 \quad 6}$	
4	$\boxed{3 \quad 1}$	
5	2	5
6	1	3



M2
⇓
2-states

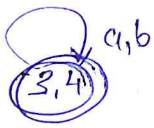


unreachable state

Eg 4:

M1

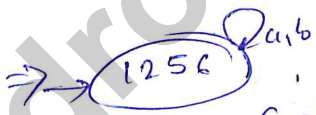
	a	b
→1	6	6
2	5	2
3	$\boxed{4 \quad 4}$	
4	$\boxed{3 \quad 3}$	
5	2	5
6	1	1



unreachable state

$L = \{ \} = \phi$

1,2,5,6

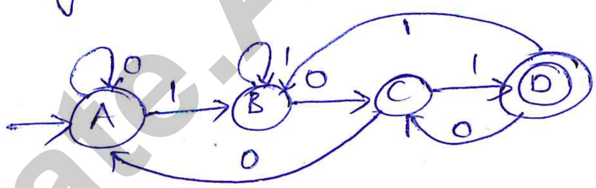


M2 Contains only one state

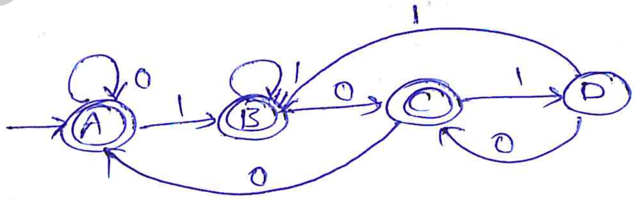
$L = \{ \} = \phi$

Eg 5: Build min FA that does not accept strings ending in 101 $\Sigma = \{0,1\}$

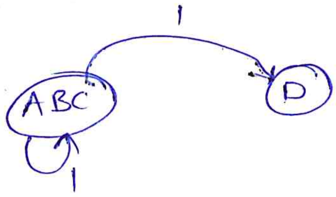
M:

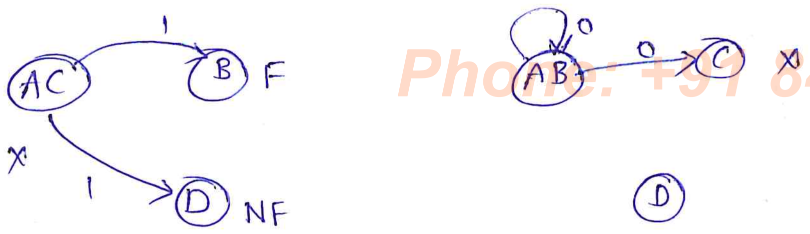


M':

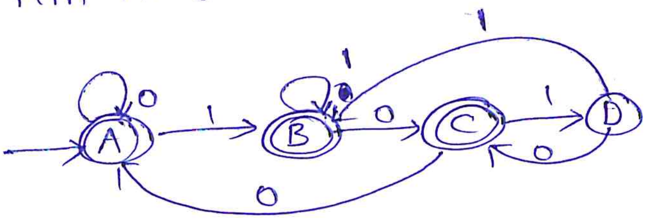


×





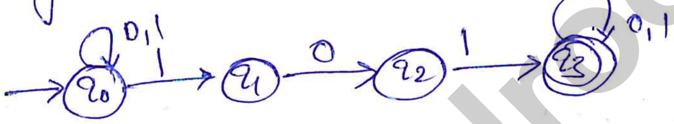
⇒ Min FA is



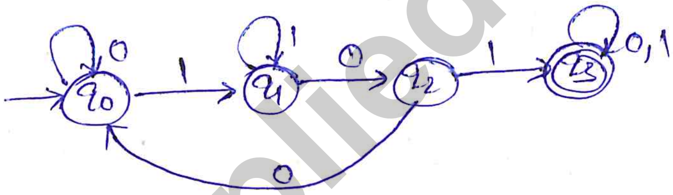
Contains 4 States.

<u>Eg 6:</u> Build min FA that accepts string that contain 101 as substring.

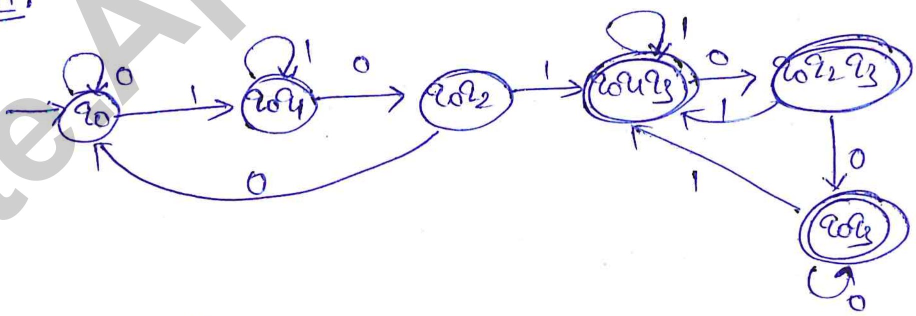
<u>NFA:</u>



<u>Min DFA:</u>

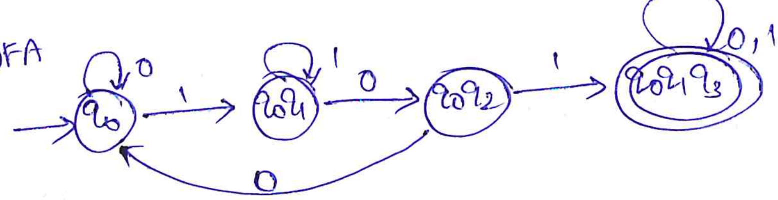


<u>NFA to DFA:</u>



⇓

Min DFA



Contains 4 States

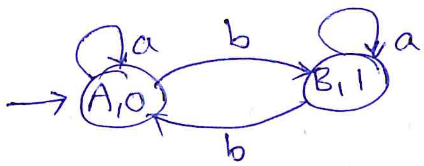
Moore Machine + solved problems:

FA with outputs:

↳ DFA, NFA, ϵ-NFA ⟶ acceptance [Final States] ← without output

* Moore Machine
* Mealy Machine

Moore Machine:



$\Sigma = \{a, b\}$ ← input set

$\Delta = \{0, 1\}$ ← output set

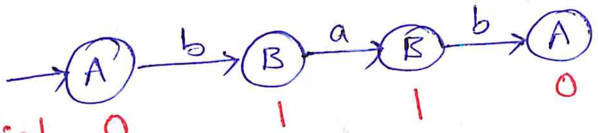
$Q = \{A, B\}$ ← states

δ: Transition Function

λ: output function

no Final states

let $W = bab$



initial State

input: n-length word / string

output: (n+1) length

$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ ⟶ 6-Tuple

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

$\lambda: Q \to \Delta$

Note:- let $W = \epsilon$ then output $= \lambda(q_0)$

DFA, NFA, ϵ-NFA ⟶ Transition Graph (or) table

δ	a	b	output
A	A	B	0
B	B	A	1

(Counter)
↳ S/W or H/W

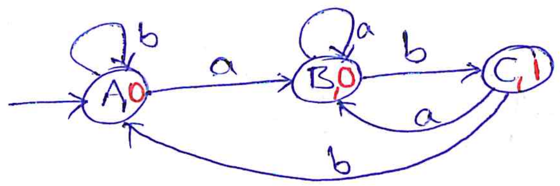
let w = abab

0 0101 → count the number of 1's = 2 → # of b's in the input string.

Not a binary number ← 0 0101

Eg 2: Construct a moore machine that counts # of occurancy of substring 'ab'. $\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$

DFA:

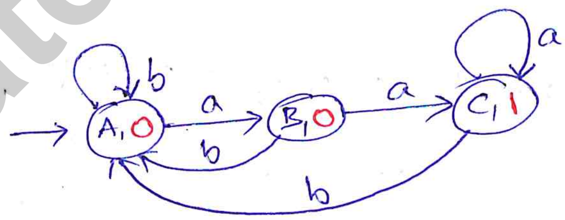


ab → 001

aab → 0001

abab → 00101

Eg 3: Counts # occurancy of two consecutive a's.



aa → 001

baa → 0001

bbbb → 00000

aaa → 0011

aab → 0010

Eg 4: Output 1's complement of a binary input string. $\Sigma = \{0, 1\}$ $\Delta = \{0, 1\}$

1011

0100 ← 1's Complement

H/W

let input String W = 10

$A \xrightarrow[0]{1} A \xrightarrow[1]{0} B$ output

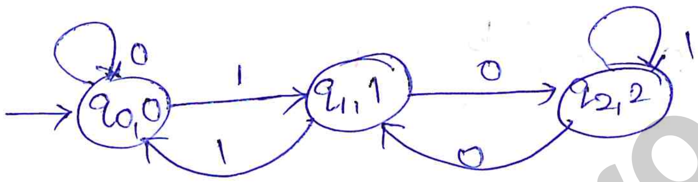
W = 0 1

	0	1
↓	↓	↓
~~0~~	1	0

ignore

Eg 5: Input : Binary String, output = (input) mod 3

$\Sigma = \{0, 1\}$ $\Delta = \{0, 1, 2\}$



bin	dec	Rem
0000	0	0
0001	1	1
0010	2	2
0011	3	0
0100	4	1
0101	5	2
0110	6	0

Let W = 1 0 1

↑ ↑ ↑ ↑

0 1 2 2 ⇒ is the output

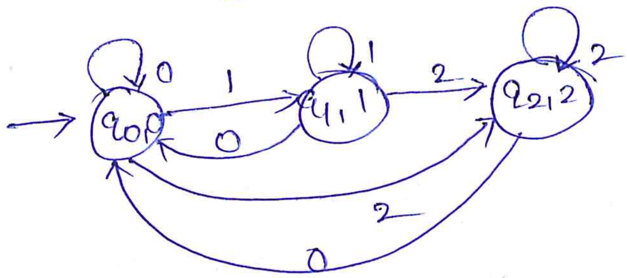
ignore

= (101) mod 3

= 2

Eg 6: input: base 3 number ; output = $(\text{input})_3$ mod 3

$\Sigma = \{0, 1, 2\}$ $\Delta = \{0, 1, 2\}$



$(120)_3 = 0 + 6 + 9 = 15$

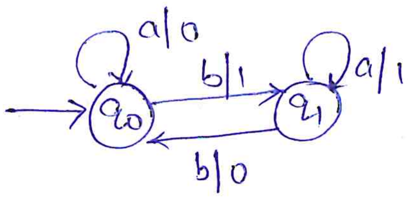
base3	dec	mod3
0	0	0
1	1	1
2	2	2
10	3	0
11	4	1
12	5	2
120	15	0
121	16	1
122	17	2

$W = \;\; 1 \;\; 2 \;\; 2$

$\uparrow \uparrow \uparrow \;\; \uparrow$

$\underbrace{0 \;\; 1 \;\; 2}_{\text{ignore}} \;\; \boxed{2} \rightarrow$ output

Mealy Machine :- Output is associated with the transition



$Q = \{q_0, q_1\}$

$q_0 = q_0$

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

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

$\overbrace{W = a\,b\,a\,b\,a}^{n}$

$\uparrow \uparrow \uparrow \uparrow \uparrow$

$\underbrace{0 \;\; 1 \;\; 1 \;\; 0 \;\; 0}_{n}$

$Q = \{q_0, q_1\}$

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

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

$q_0 = q_0$

$\delta : Q \times \Sigma \rightarrow Q$

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

$\Rightarrow$ $\left.\begin{array}{l} W = \epsilon \\ \text{output} = \epsilon \end{array}\right\}$

$M = (Q, \Sigma, \delta, q_0, \Delta)$

→ Transition diagram

→ Transition Table

(1)

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_0

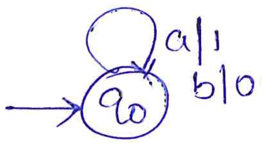
λ	a	b
q_0	0	1
q_1	1	0

(2)

	a	b
q_0	q_0, 0	q_1, 1
q_1	q_1 1	q_0 0

Eg 1: Count # occurances of a's in a String $\Sigma = \{a, b\}$

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

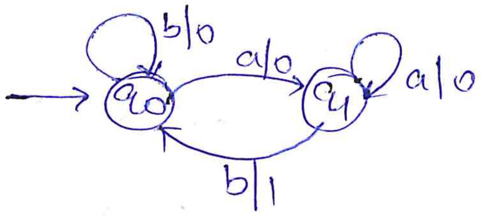


$\Rightarrow$ $w = ababa$

$\Rightarrow$ $q_0 \xrightarrow[1]{a} q_0 \xrightarrow[0]{b} q_0 \xrightarrow[1]{a} q_0 \xrightarrow[0]{b} q_0 \xrightarrow[1]{a} q_0$

Moore $\longrightarrow$ Mealy

Eg 2:- Count the number of occurancy of 'ab' as substring



Let $w = abab$

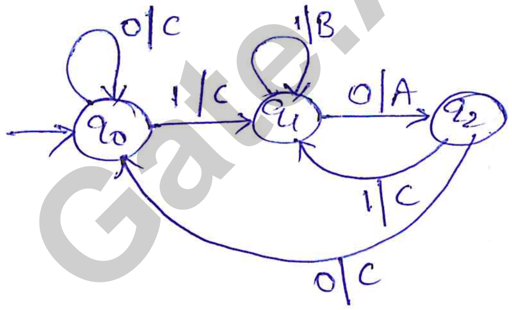
$\uparrow\uparrow\uparrow\uparrow$
0 1 0 1

Eg 3: $\Sigma = \{0, 1\}$ $\Delta = \{A, B, C\}$

$x10$: A

$x11$: B

otherwise : C



1 : C

0 : C

10 : CA

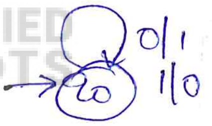
11 : CB

110 : CBA

101 : CAC

1011 : CACB

Eg 4:- 1's Complement Mealy Machine

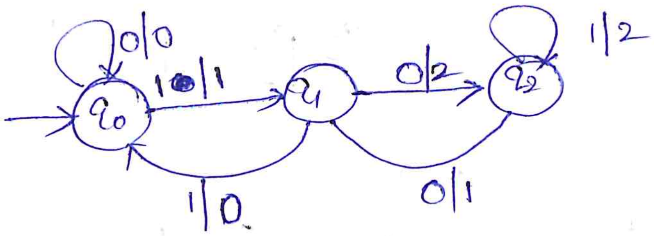


$\Sigma = \{0,1\}$

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

Eg 5: $\Sigma = \{0,1\}$. $(\text{input})_2 \bmod 3$

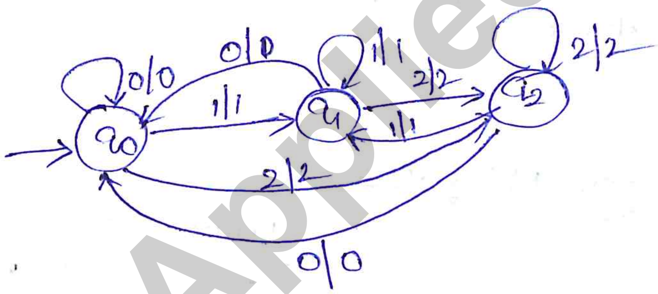
$\Sigma = \{0,1\}$
$\Delta = \{0,1,2\}$



bi	Dec	Mod3
0	0	0
1	1	1
10	2	2
11	3	0
100	4	1
101	5	2
110	6	0

Eg 6: $\Sigma = \{0,1,2\}$ output $= (\text{input})_3 \bmod 3$

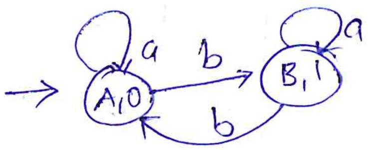
$\Delta = \{0,1,2\}$



Base3	Dec	Mod3
0	0	0
1	1	1
2	2	2
10	3	0
11	4	1
12	5	2
120	15	0
121	16	1
122	17	2

Conversions: Moore → Mealy Machine

Eg 1:



δ	a	b	λ
A	A	B	0
B	B	A	1

→

Mealy:

δ	a o/p	b o/p
A	A 0	B 1
B	B 1	A 0

Eg 2:-

Moore

σ	a	b	λ
→q_0	q_1	q_2	0
q_1	q_0	q_3	1
q_2	q_3	q_0	0
q_3	q_2	q_1	1

Mealy

δ, λ	a	b
→q_0	q_1 1	q_2 0
q_1	q_0 0	q_3 1
q_2	q_3 1	q_0 0
q_3	q_2 0	q_1 1

Eg 3:

Moore

σ	a	b	λ
→q_0	q_2	q_4	0
q_1	q_0	q_2	1
q_2	q_3	q_1	2
q_3	q_4	q_0	0
q_4	q_1	q_3	1

Mealy

σ	a	b
→q_0	q_2 2	q_4 1
q_1	q_0 0	q_2 2
q_2	q_3 0	q_1 1
q_3	q_4 1	q_0 0
q_4	q_1 1	q_3 0

states don't change.

Mealy M/c to Moore Machine:

Eg 1:

δ, λ	a	b
→q_0	$\boxed{q_2, 0}$	$\boxed{q_1, 0}$
q_1	$q_0, 0$	$\boxed{q_2, 1}$
q_2	$\boxed{q_1, 1}$	$q_0, 0$

$q_0 \rightarrow 0$

$q_1 \rightarrow 0, 1$

$q_2 \rightarrow 0, 1$

→ **Moore M/c**

σ	a	b	λ
q_0	q_{20}	q_{10}	0
q_{10}	q_0	q_{10}	0
q_{11}	q_0	q_{10}	1
q_{20}	q_{11}	q_0	0
q_{21}	q_{11}	q_0	1

states need not be the same.

Eg 2:

Mealy

δ, λ	0	1
$\rightarrow q_0$	$q_2, 1$	$q_3, 1$
q_1	$q_3, 1$	$q_4, 1$
q_2	$q_0, 1$	$q_1, 0$
q_3	$q_4, 0$	$q_2, 0$
q_4	$q_1, 0$	$q_0, 0$

$q_0 <^{0}_{1}$

$q_1 \rightarrow 0$

$q_2 <^{0}_{1}$

$q_3 - 1$

$q_4 <^{0}_{1}$

Moore Machine :-

δ	0	1	λ
(81) $\rightarrow q_{00}$	q_{21}	q_3	0
$\rightarrow q_{01}$	q_{21}	q_3	1
q_1	q_3	q_{41}	0
q_{20}	q_{01}	q_1	0
q_{21}	q_{01}	q_1	1
q_3	q_{40}	q_{20}	1
q_{40}	q_1	q_{20}	0
q_{41}	q_1	q_{00}	1

Regular Languages:

FA $\rightarrow$ DFA, NFA, ϵ-NFA
FA $\rightarrow$ Moore, Mealy

A Language is regular iff
$\exists$ a FA that accepts every word in L.

L is RL if $\exists$ FA, M such that $L(M) = L$

Automata	Lang
FA $\longleftrightarrow$	RL
PDA $\longleftrightarrow$	..
TM $\longleftrightarrow$	..

→ Non-regular Language - No FA exist

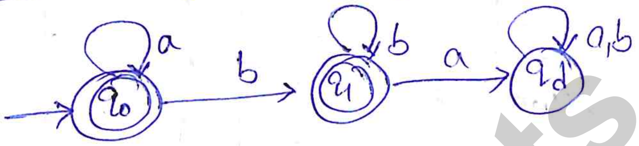
→ Every finite Language is Regular

→ $L = \phi = \{\}$ Regular

Examply:

① $L = \{ab, abb, aab\}$ Finite - Regular

② $L = \{a^m b^n \mid m, n \geq 0\}$ Not finite - Regular



③ $L = \{a^m b^n \mid m \geq 2, n \geq 3\}$ - Regular

L → F, NF
F → R
NF → R, NRL

④ $L = \{a^m b^n \mid m*n = \underline{\text{Const}}\}$ - Finite (10)
Finite Regular

⑤ $L = \{a^m b^n \mid m = n = \text{Const}\}$ - Finite - Regular (20)

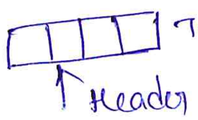
⑥ $L = \{a^m b^n \mid 1 \leq m = n \leq \text{Const}\}$ - Finite - Regular (5)
$\{ab, aabb\}$

⑦ $L = \{a^m b^n \mid 1 \leq m \leq n \leq \text{Const}\}$ - Finite - Regular (10)

⑧ $L = \{a^n b^n \mid n \geq 1\}$: Non-finite = $\{ab, aabb, \ldots\}$
↳ Non-Regular   pumping lemma

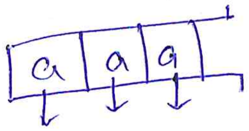
× States in FA to
Keep a count → ∞

→ Variable: cnt (increment) → read/write



Tape is read only

→ Stack: For all a's push operation, for every b apply one pop operation.



FA + Stack → PDA (push down automata)

FA + read/write capability : TM

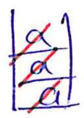
(9) $L = \{a^m b^n \mid m = 10, 20, 30, \ldots;\ n = 5, 10, 15, \ldots\}$

Reminder
$m \pmod{10} = 0$ $(0 \ldots 9)$
$m \pmod 5 = 0$ $(0 \ldots 4)$

⇓

Regular

(10) $L = \{a^m b^n \mid m < n\}$ → Non-Finite → Non-Regular



$\cancel{b}\cancel{b}\cancel{b}b$
Skip

Stack

(11) $L = \{a^m b^n \mid m > n\}$ → Non-Regular

(12) $L = \{a^m b^n \mid m \neq n\}$ → Non-Regular

(13) $L = \{a^m b^n \mid m = 2n\}$ → Non-Regular

(14) $L = \{a^m b^n \mid m = n^2\}$ → Non-Regular

(15) $L = \{a^m b^n \mid m + n = 10\}$ → Finite → Regular

(16) $L = \{a^m b^n \mid \gcd(m, n) = 1\}$ = Store Reminder
⇓
NRL

(17) $L = \{a^m b^n \mid m + n = \text{even}\}$ → Regular

(18) $L = \{w w^R \mid w \in \Sigma^*\}$ → Non-Regular

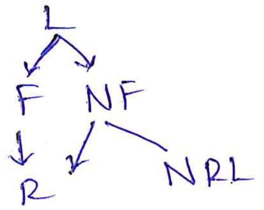
(19) $L = \{ w \in \Sigma^* \mid |W_a| = |W_b| \}$ - Not Finite $\Downarrow$ Non-Regular

$\Rightarrow$ abab

Properties of RL:

(1) Finite Language $\Rightarrow$ RL

(2)



(3) RL: Finite or Non-Finite

(4) Every NRL has to be infinite Lang

(5) Subset of a RL, need not be regular

$RL\ L_1 \Rightarrow \{ a^m b^n \mid m, n \geq 0 \}$

$L_2 = \{ a^m b^n \mid m = n \geq 0 \}$

$\boxed{L_2 \subset L_1}$

(6) Every finite subset of a RL is a RL

(7) Every finite subset of a NRL is a RL.

(8) ~~Every~~ Subset of a NRL need not be a NRL

$L = \{ a^m b^n \mid m = n \}$ NRL

$L' = \{ ab, aabb, aaabbb \}$ RL

(9) Supersets of a RL need not be RL

" " NRL " " NRL

(10) Finite union of Regular Languages is a RL

$\underset{M}{L} = \underset{m_1}{L_1} \cup \underset{m_2}{L_2} \cup \underset{m_3}{L_3} \cdots \cup \underset{m_n}{L_n}$

Infinite union of RL need not be RL

(11) Finite intersection of RL is a RL

$$L = L_1 \cap L_2 \cap L_3 \cdots L_n$$

$$\uparrow$$

$$M \Rightarrow M_1 \quad M_2 \quad M_3 \cdots M_n$$

Infinite intersection of RL need not be RL.

Regular Expressions :- (RE)

GREP, Python, Java - String matching

→ One way to represent a RL

$L = \{ \cdots \}$

FA

→ operators : $*$, $\cdot$, $+$ widely used operators symbols in RE

$*$ → Kleene closure

$\cdot$ → Concatenation

$+$ → OR union

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

(1) $r = a^* = \{\epsilon, a, aa, aaa, \cdots\}$

(2) $r = a + b = \{a, b\}$

(3) $r = ab = \{ab\}$

(4) $a^* + ba = \{\epsilon, a, aa, \cdots, ba\}$

(5) $(ab)^* + b^* a^* =$

$\{\epsilon, ab, abab, \cdots\} \cup$

$\{\epsilon, b, bb, \cdots\}\{\epsilon, a, aa, \cdots\}$

$\Rightarrow \{\epsilon, a, aa, b, ba, baa, bb, bba \cdots\}$

Order of precedence:

eg: $(a+ba)^*$ = {∈, a, ba, aa, baba .....}

1, 2, 3

First : *

Second : .

Third : +

$(a+ba)^* = \epsilon \cup (a+ba) \cup (a+ba)^2 \ldots$

$(a+ba)^2 = (a+ba)(a+ba)$

= {aa, aba, baa, baba}

$r = a^+ = \{a, aa, aaa, \ldots\}$

Types of RE:

- Restricted / Standard RE $(*, \cdot, +)$
- Semi-restricted $(*, \cdot, +, \cap)$
- un-restricted RE $(*, \cdot, +, \cap, \sim)$

Observations:

① Every finite Language is Regular & we can express it using a Regular expression.

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

↳ $w_1 + w_2 \ldots + w_n$

reg expr ⟷ FA

ab + aba + ba + abb

② $r^* = \{\epsilon, r, rr, \ldots\}$

$r^+ = \{\bullet, r, rr, \ldots\}$

$L(r^*) = L(r)^*$

⇒ r = ab

⇒ $L((ab)^*) = \{\epsilon, ab, abab, \ldots\}$

$\Rightarrow (L(r))^*$

$\Rightarrow (ab)^* = \{\epsilon, ab, abab, \ldots\}$

(3) $r = \epsilon \Rightarrow r^* = \{\epsilon, \epsilon\epsilon, \epsilon\epsilon\epsilon \ldots\}$

$= \{\epsilon\}$

(4) $r = \epsilon \Rightarrow r^+ = \{\epsilon, \epsilon\epsilon, \ldots\} = \{\epsilon\}$

(5) $r = \phi = \{\} \Rightarrow r^* = \{\epsilon\}$

$\Rightarrow r^+ = \{\ \} = \phi$

(6) $r^+ \subset r^*$

$r^* = r^+ \cup \{\epsilon\}$

(7) $(r^*)^* = r^*$

$r^* \cdot r = \{r, r^2, r^3, \ldots\} = r^+$

$r^3 \rightarrow r \cdot r \cdot r$

Regular Expressions : Solved problems

(1) Which is correct

(a) $r^* = r(*)$ ✗

(b) $(r^*)^+ = r^+$ ✗ $= \{r, rr, \ldots\}$

LHS $\{r^*, r^*r^*, \ldots\}$

$\hookrightarrow \epsilon, r, rr \ldots$

(c) $(r^*)^* = r^*$ ✓

$\Downarrow$

LHS $= \{\epsilon, r^*, r^*r^*, \ldots\}$

$= \{\epsilon, r^*, r^* \ldots\}$

$= r^*$

(d) $(r^+)^* = r^+$ ✗

$\downarrow$

$\{\epsilon, r^+, r^+r^+, \ldots\}$

Q2 which of the following is/are correct.

✗ (a) $r^* = r^+$

✓ (b) $r^* = \bigcup_{i \geq 0} r^i = r^0 \cup r^1 \cup r^2 \ldots$

✓ (c) $r^+ = \bigcup_{i \geq 1} r^i = r^1 \cup r^2 \cup r^3 \ldots$

✓ (d) $r^* = r^+$ iff $r = \epsilon$

Q3 which of the following is/are incorrect

✓ (a) $r^* + r^+ = r^*$ LHS $= \{\epsilon, r, r^2 \ldots\} \cup \{r, rr \ldots\} =$ RHS

✓ (b) $r^* \cdot r^+ = r^+$

✗ (c) $r^* \cdot r^* = r^+$ – Answer

✓ (d) $r^+ (r^*)^+ = r^+$

Q4 Choose the correct statement

✗ (a) r^*, r^+ always represent ∞-lang $r = \epsilon$ $r^* = \{\epsilon\}$ $= r^+$

✗ (b) r^*, r^+ are always finite language $r = a$ $r^* = \{\epsilon, a, aa \ldots\}$

✓ (c) r^*, r^+ are finite iff $r = \phi$ or $r = \epsilon$

✗ (d) None

Q5 which of the following are identical

a) $a^* = \{\epsilon, a, aa, aaa \ldots\}$ $a \neq b$

b) $(aa)^* = \{\epsilon, aa, aaaa, \ldots\}$

c) $(aa^*)a = \{aa, \ldots\}$ $a \neq c$ $b \neq c$

d) $(a+\epsilon)a^* = \{\epsilon, a, aa, aaa \ldots\}$

$\boxed{a = d}$ $a^* = (a+\epsilon)a^* \leftarrow$ Reg lang : Multiple RE that can generate it

RE $\rightarrow$ only one RL

Q6 which is correct

✗ a) $r_1^* + r_2^* \neq (r_1 + r_2)^*$

✗ b) $r_1^* r_2^* \neq (r_1 + r_2)^* \longrightarrow r_2 r_1$

✓ c) $(r_1^* + r_2)^* = (r_1 + r_2)^*$

✗ d) $(r_1 r_2)^* \neq (r_1 + r_2)^*$

$\downarrow$ $r_1 r_2$ $\quad r_2 r_1$

LHS: $r_1 r_1$, $r_2 r_2$

RHS: $r_1 r_2$, $r_1 r_2 r_1$

$r_1 = a$, $r_2 = b$ } aba

Q7 let $r_1 = 0^*1$ & $r_2 = (0^*1^*)^*$ which is correct $\Sigma = \{0,1\}$

✗ a) $L(r_1) = L(r_2)$

✓ b) $L(r_1) \subset L(r_2)$

✗ c) $L(r_1) \supset L(r_2)$

d) None

0^*1

$\{1, 01, 001, 0001, \ldots\}$

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

$\{\epsilon, 0, 1, 01, 001, \ldots\}$

Q8) $r = (a^*b)^*$ $S = (a+b^*)^*$ which of the following is correct

✗ (a) $L(r) = L(S)$

✓ (b) $L(r) \subset L(S)$

✗ (c) $L(r) \supset L(S)$

(d) None

$L(r) = \{\epsilon, a, b, ab, aab, \dots\}$

$L(S) = \{\epsilon, a, b, ab, aab, \dots\}$

$ba \neq L(r)$

$ba = L(S)$

Q9) which of the following does not contain the string 1011

✓ (a) $0^*(10)^*1$ 1010101

(b) $(0^*+1^*)^*$ 1011

✓ (c) $(110)^*0^*1$

✗ (d) $0^*(101)^*1$

Answer: a and c

Q10) which of the following REs does not generate string containing a substring '100'.

✓ (a) $0^*(10)^*$ = 000 . . 101010 . . .

✗ (b) $0^*1^*(10)^*0$ = 10100

✗ (c) $(01^*+0)^*(01)^*1^*0^*$ = 0100 . . . .

✗ (d) $(110^*+1)^*0$ = 11000 . . . .

Q11) which of the following REs does not generate string that contains 1101 as substring

✗ (a) $(101^*)^*01^*(01)^*$ 10110 1 . . . .

✗ (b) $(0^*+1^*)^*(11)^*$ 1101 . . .

× (c) $(0^*1+11)^* 01^*$ 110111

× (d) $(11^*0)^*(01)^*$ 110110

(e) None

Q12) which of the following REs does not generate string that contains 110 as substring.

× (a) $(0^*1^*)^*$ 011011

× (b) $(1^*01)^* 0$ 1101

× (c) $0^*(11)^* 0^* 1^*$ 1101

✓ (d) $(01)^* 0^* 1$ 010101000

Q13) Is $(r_1 r_2)^* r_1 = r_1 (r_2 r_1)^*$? Shifting Rule of RE

$r_1 r_2 r_1 r_2 \ldots r_1$

$\Rightarrow 0 \quad \epsilon r_1 = r_1 \epsilon$

$1 \quad (r_1 r_2) r_1 = r_1 (r_2 r_1)$

$2 \quad r_1 r_2 r_1 r_2 r_1 = r_1 r_2 r_p r_2 r_1$

$3 \rightarrow \vdots \qquad \vdots$

Note: $(r_1 + r_2)^* = (r_1^* + r_2^*)^*$

$= (r_1^* + r_2)^*$

$= (r_1 + r_2^*)^*$

$= (r_1^* \cdot r_2^*)^*$

$(r_1 + r_2)^* = (r_1^* \cdot r_2^*)^*$

$\boxed{r_1 r_2 r_1}$ $\boxed{r_1 r_2 r_1}$

Powers of Alphabet:

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

$\Sigma^1 = \Sigma = a+b \leftarrow$ RE

$\Sigma^2 = \Sigma \cdot \Sigma = (a+b)(a+b) = (a+b)^2$

$\Sigma^3 = \Sigma \cdot \Sigma \cdot \Sigma = (a+b)(a+b)(a+b) = (a+b)^3$

$\vdots$

$$\boxed{\Sigma^* = (a+b)^*}$$

$$\boxed{\Sigma^+ = (a+b)^+}$$

Constructing RE:

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

1. (a) RE that generates all strings including ϵ

$r = (a+b)^*$

(b) All strings excluding ϵ

$r = (a+b)^+$

2. Strings that start with 0 $\quad \Sigma = \{0, 1\}$

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

3. Strings that start with '10'

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

4. Start with 0110

$r = 0110(0+1)^*$

5 (a) Strings that end in '0'

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

(b) Strings that end in '10'

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

(c) Strings that end in '1001'

$r = (0+1)^* 1001$

6 (a) Contains substring '10'

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

— 10 —

(b) Contains substring '010'

$r = (0+1)^* 010 (0+1)^*$

7 (a) Strings starts & ends in 0

$0 \, 111 \, 0$

$w = 0$

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

7 (b) Strings starts & ends in same symbol

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

7 (C) Strings that start & end in different symbol

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

(8) (a) 3rd symbol from left is 1

$$\underline{(0+1)}_1\ \underline{(0+1)}_2\ \underline{1}_3\ \underline{\ }_4\ \underline{\ }_5\ \underline{\ }_6 \ldots \Rightarrow (0+1)^2\, 1\, (0+1)^*$$

(b) 3rd symbol from the right is 0

$$-\ \underset{3}{\underline{0}}\ \underset{2}{\underline{\ }}\ \underset{1}{\underline{\ }}$$

$$(0+1)^* 0 (0+1)^2$$

(9) (a) Strings that contain exactly Two 0's. $\Sigma = \{0,1\}$

$$r = 1^* 0 1^* 0 1^*$$

(b) atmost two 0's

$$r = 1^*(\epsilon + 0)\, 1^* (0+\epsilon)\, 1^*$$

(c) atleast two 0's

$$r = (0+1)^* \underline{0} (0+1)^* \underline{0} (0+1)^*$$

(d) # zero's $\equiv 0 \pmod 3$ 0, 3, 6, 9, 12 . . .

$$\underset{0}{1^*} + (\underbrace{1^* 0 1^* 0 1^* 0 1^*}_{3})^*$$

(e) # 1's $\equiv 2 \pmod 3$ 2, 5, 8, 11, . . .

$$r = \underbrace{0^* 1 0^* 1 0^*}_{2} (\underbrace{0^* 1 0^* 1 0^* 1}_{3})^*$$

(10) (a) $|w| = 2$ $\Sigma = \{0,1\}$

$$\underline{(0+1)}\ \underline{(0+1)} = (0+1)^2$$

10 (b) $|w| \le 2$

$$\underline{(\epsilon+0+1)}\,\underline{(\epsilon+0+1)} = (0+1+\epsilon)^2$$

(c) $|w| \ge 2$

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

$$\Downarrow$$

$$\underbrace{(0+1)^2}_{2}\,\underbrace{(0+1)^*}_{\{0,1,2\ldots\}}$$

(d) $|w| \equiv 0 \pmod 2$ $\quad 0, 2, 4, 6, \ldots$

$\downarrow 2\times 0 \quad \downarrow 2\times 1$

$$((0+1)^2)^*$$

(e) $|w| \equiv 1 \pmod 2$ $\quad 1, 3, 5, \ldots$

$$(0+1)((0+1)^2)^*$$

(f) $|w| \equiv 2 \pmod 3$ $\quad 2, 5, 8, 11 \ldots$

$$\underbrace{(0+1)^2}_{2}\,\underbrace{((0+1)^3)^*}_{\{0,3,6,\ldots\}}$$

11 (a) $L = \{0^n \mid n \ge 0\} \Rightarrow 0^*$ $\quad \Sigma = \{0\}$

(b) $L = \{0^n \mid n \ge 1\} \Rightarrow 0^+$ $\quad \Sigma = \{0\}$

(c) $L = \{1^n \mid n \ge 3\} \Rightarrow$ $\quad \Sigma = \{1\}$

$$1111^* \Rightarrow 111^+$$

(d) $L = \{0^m 1^n \mid m, n \ge 0\} \Rightarrow 0^* 1^*$

(e) $L = \{0^m 1^n \mid m \ge 1, n \ge 1\} \Rightarrow \boxed{0^+ 1^+}$ or $\boxed{00^* 11^*}$

(f) $L = \{0^m 1^n \mid m \geq 0,\ n \geq 1\}$ $\quad r = 0^*11^* \Rightarrow 0^*1^+$

(g) $L = \{0^m 1^n \mid m \geq 2,\ n \geq 3\}$ $\quad r = 000^*\,1111^*$

(h) $L = \{0^m 1^n 2^p \mid m \geq 0,\ n \geq 1,\ p \geq 2\}$ $\quad 0^*11^*222^*$

$\Downarrow$

$0^*1^+22^+$

(12) $L = \{0^m 1^n \mid m+n = \text{even}\}$

$m = \text{even } 2x \quad m = \text{odd } 2x+1 \quad x \geq 0$

$n = \text{even } 2x \quad n = \text{odd } 2x+1$

$0^{2x}1^{2x} \qquad 0^{2x+1}1^{2x+1}$

$(00)^x(11)^x \qquad (00)^x0\,(11)^x1$

$\Downarrow$

$\boxed{(00)^*(11)^* + (00)^*0(11)^*1}$

(13) $L = \{0^m 1^n \mid m+n = \text{odd}\}$

$m = \text{odd}$ (or) $m = \text{even}$

$n = \text{even}$ $\quad n = \text{odd}$

$2x+1 \qquad 2x$

$2x \qquad 2x+1$

$\boxed{(00)^*0(11)^* + (00)^*(11)^*1}$

(14)(a) strings start with '0' & $|w| = \text{even}$ $\qquad \boxed{0\ \text{odd}} \Rightarrow \text{even}$

odd

$r = 0(0+1)((0+1)^2)^*$

(b) Starts with '1' & $|w|$ = odd

$r = 1((0+1)^2)^*$

$|1 \underbrace{\quad}_{\text{even}}| = \text{odd}$

(15) Starts with 0 & do not contain two consecutive ones.

$r = 0$

(11)

$\times\ ((00)^* + (01)^* + (10)^*)^*$

$\underline{000110}$

$\{\epsilon, 00, 0000 \cdots\} (00)^*$

$\{\epsilon, 01, 0101 \cdots\} (01)^*$

$(10)^*$

$\{\epsilon, 10, 1010, \cdots\} \Rightarrow (11)^*$

$0^+ = \{0, 00, \cdots\}$

$(01)^+ = \{01, 0101, \cdots\}$

$(0+01)^{*+}$

$\underline{0}\ \ \underline{01}$

(16) Strings do not contain Two consecutive 0's & 1's.

$(01)^* = \{\epsilon, 01, 010101, \cdots\}$

$(10)^* = \{\epsilon, 10, 1010 \cdots\}$

$r = (1+\epsilon)(01)^*(0+\epsilon)$

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

(17) String contains exactly two consecutive 1's.

$r = 0^* 110^*$

exactly 2 consecutive 1's.

$0^* 1 0^* 1 0^*$

exactly 2 1's

(18) # Strings of length ≤ 3 generated by $r = (1+01)^*0$

len = 0 → { }

len = 1 → {0} — ①

len = 2 → {10} — ①

len = 3 → {110, 010} — ②

4 distinct strings.

(19) # distinct strings of length ≤ 3 generated by $r = (0+01)^*1\,(0+1)^*$

$(0+01)^*1(0+1)^*$

len = 0 { }

len = 1 {1} — ①

len = 2 {01, 10, 11} — ③

len = 3 {001, 010, 011, 100, 101, 110, 111} — ⑦

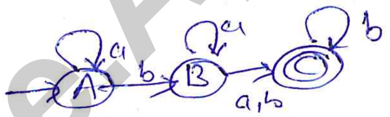
11 strings

(20) Finite Automata - Regular Expression:

FA ⟷ RE

NFA, DFA

Eg 1:



$A = \epsilon + Aa$ — ①

$B = Ab + Ba$ — ②

$C = B(a+b) + Cb$ — ③

Arden's Lemma/Rule

let R, P, Q → reg Expressions

$R = Q + RP$ → linear Equation in R

$X = B + XA \Rightarrow AX + B = X$

Solution is

$R = QP^*$, if P does not contain ϵ then

QP^* is the unique solution.

else $R = Q + RP$ has ∞-many solutions

$A = \epsilon + Aa$

$\Rightarrow A = \epsilon a^*$

$B = Ab + Ba$

$= a^* b + Ba$

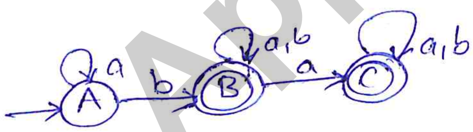
$\Rightarrow B = a^* b a^*$

$C = B(a+b) + Cb$

$C = a^* b a^* (a+b) + Cb$

$\Rightarrow \boxed{C = a^* b a^* (a+b) b^*}$

Eg 2:



$A = \epsilon + Aa \Rightarrow A = a^*$

$B = Ab + B(a+b) = a^* b + B(a+b)$

$R \quad Q \quad RP$

$= \boxed{a^* b (a+b)^*}$

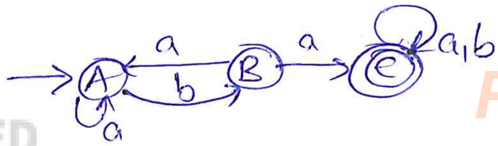
$C = Ba + C(a+b) = a^* b (a+b)^* a + C(a+b)$

$C = \boxed{a^* b (a+b)^* a (a+b)^*}$

B and C are Final States.

RE is $B + C$ $\quad a^* b (a+b)^* + a^* b (a+b)^* a (a+b)^*$

Eg 3:



$A = \epsilon + Aa + Ba$ —— ①

$B = Ab$ —— ②

Substitute ② in ①

$A = \epsilon + Aa + Aba$

$= \epsilon + A(a+ba)$

$A = \epsilon \cdot (a+ba)^*$

$\Rightarrow A = (a+ba)^*$

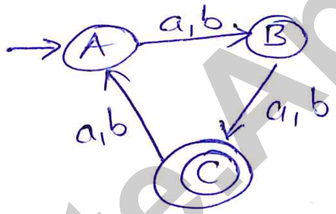
$\Rightarrow B = (a+ba)^* b$

$C = Ba + C(a+b)$

$= (a+ba)^* ba + C(a+b)$

$\boxed{C = (a+ba)^* ba\, (a+b)^*}$

Eg 4:



$A = \epsilon + C(a+b)$

$B = A(a+b)$

$C = B(a+b) = A(a+b)(a+b)$

$\Rightarrow A = \epsilon + C(a+b)$

$= \epsilon + A(a+b)^2(a+b)$

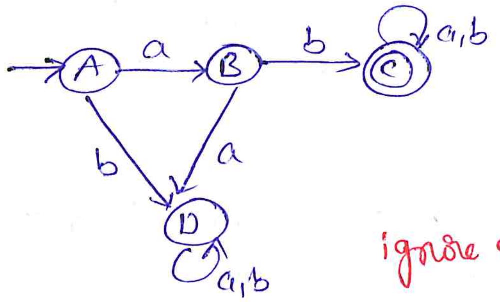
$= \epsilon + A(a+b)^3$

$\Rightarrow A = ((a+b)^3)^*$

$\Rightarrow B = ((a+b)^3)^* (a+b)$

$C = ((a+b)^3)^* (a+b)^2 \therefore$ len of String $\equiv 2 \pmod 3$

Eg 5:



ignore dead state

$A = \epsilon$

$B = Aa$

$C = Bb + C(a+b)$

$\Rightarrow A = \epsilon$

$B = a$

$C = ab + C(a+b)$

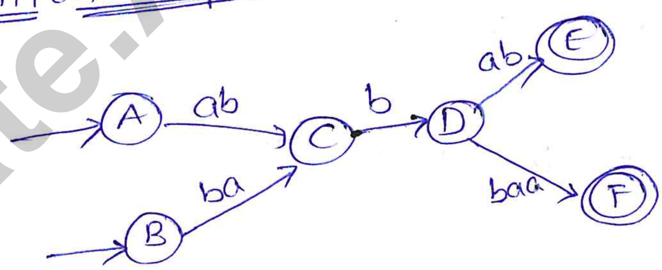
$\boxed{C = ab(a+b)^*}$

Elimination Method: (State)

ϵ-NFA $\rightarrow$ RE

Transition Graph: (TG)

NFA + multiple initial states + transition on words



$I = \{A, B\}$

$F = \{E, F\}$

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

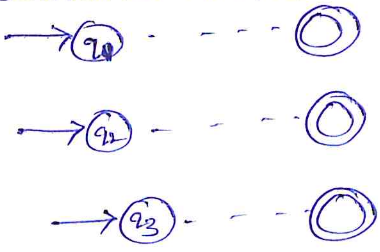
$M = (Q, \Sigma, I, F, \delta)$

$\delta : Q \times \Sigma^* \rightarrow 2^Q$

ϵ-NFA $\rightarrow$ RE
TG $\rightarrow$ RE

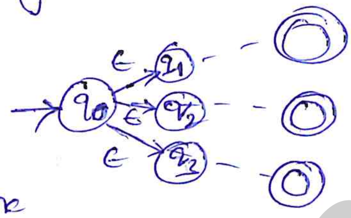
Arden's Rule
DFA / NFA $\rightarrow$ RE

Step 1: Combine initial states using ϵ-transitions



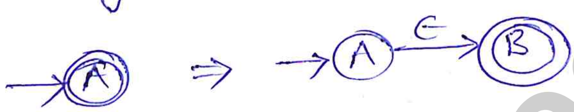
T.G

$\Rightarrow$

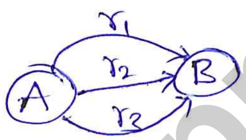


one initial state

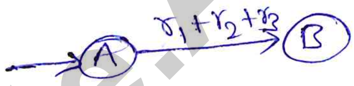
Step 2: separate out initial and final states using ϵ-transitions



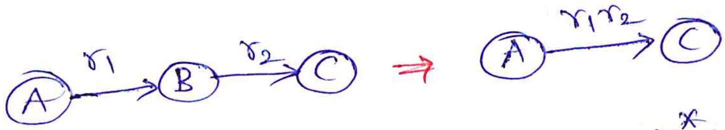
Step 3: simplifying parallel edges.

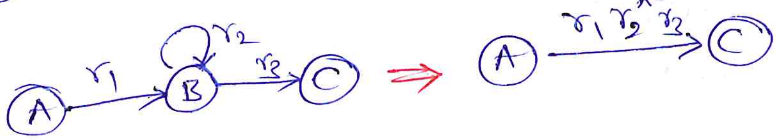


$\Downarrow$



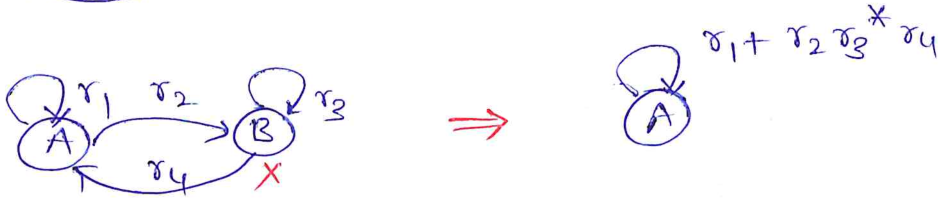
Step 4: Eliminating nodes / states

① 

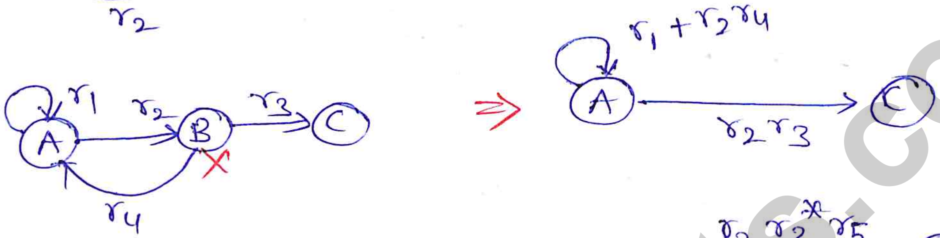
② 

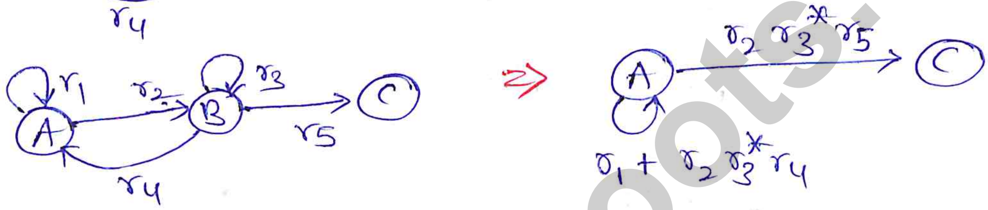
③ 

4 

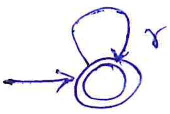
5 

6 

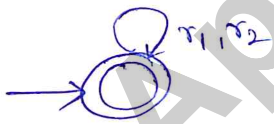
7 

8 

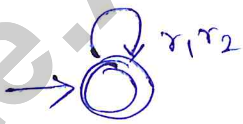
Step 5: TG is any of the following RE

1 

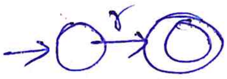
$\Rightarrow r^*$

2 

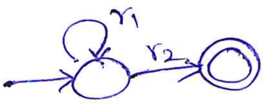
$\Rightarrow (r_1 + r_2)^*$

3 

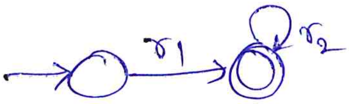
$\Rightarrow (r_1 r_2)^*$

4 

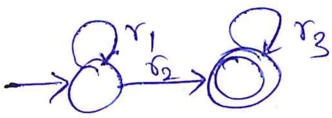
$\Rightarrow r$

5 

$\Rightarrow r_1^* r_2$

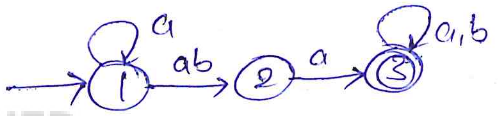
6 

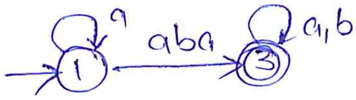
$\Rightarrow r_1 r_2^*$

7 

$\Rightarrow r_1^* r_2 r_3^*$

Q1

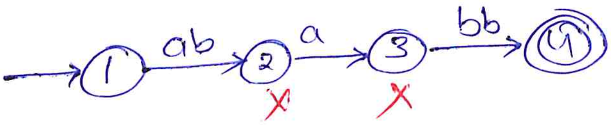


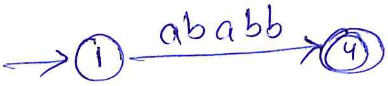


⇓

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

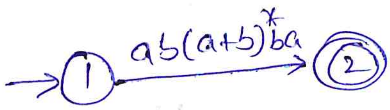
Q2





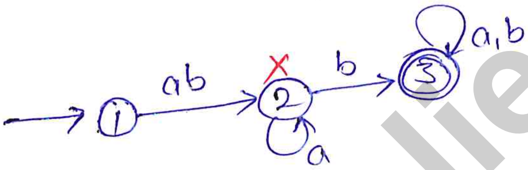
RE: ababb

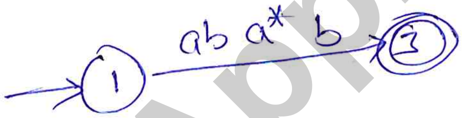
Q3



⇒ $ab(a+b)^* ba$

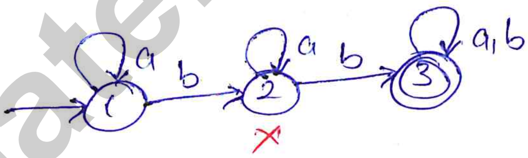
Q4

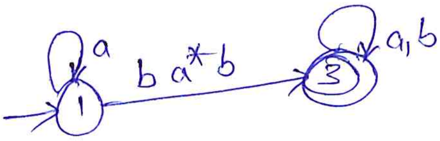




aba^*b

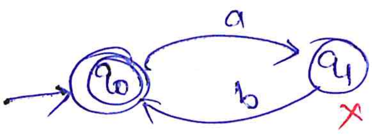
Q5

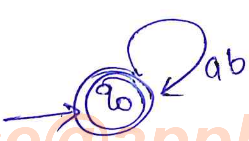




⇒ $a^* ba^*b(a+b)^*$

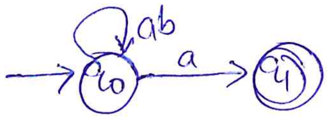
Q6





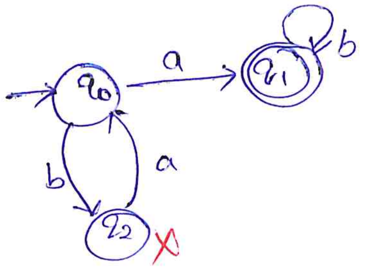
⇒ $(ab)^*$

Q7

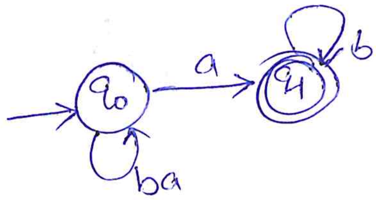


$\Rightarrow (ab)^* a$

Q8

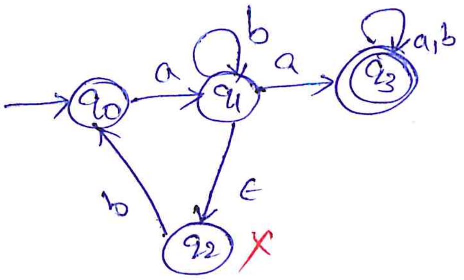


$\Rightarrow$

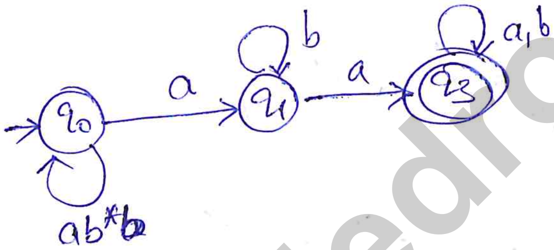


$\Rightarrow \boxed{(ba)^* a b^*}$

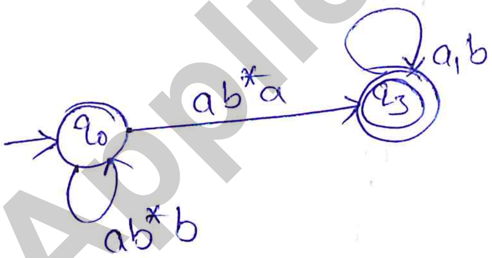
Q9



$\Rightarrow$

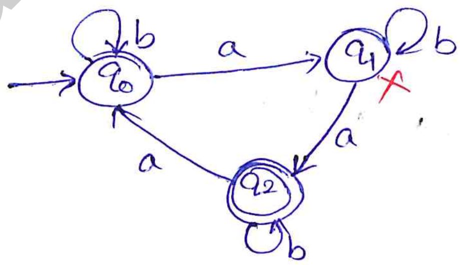


$\Rightarrow$

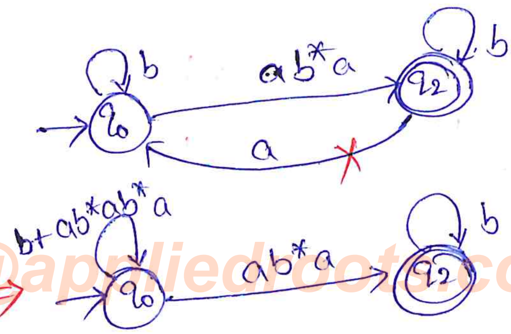


$(ab^*b)^* \; ab^*a \; (a+b)^*$

Q10

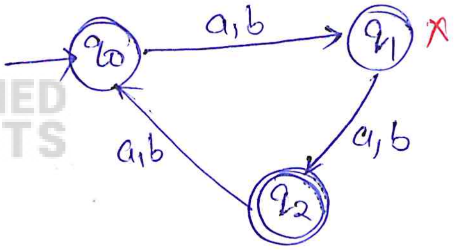


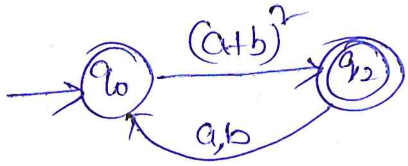
$\Rightarrow$

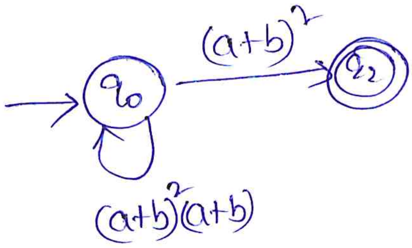


$\boxed{(b+ab^*ab^*a)^* \, ab^*a \, b^*}$

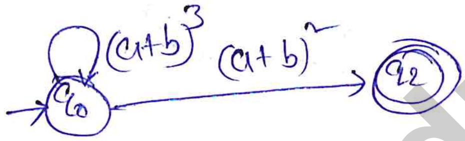
Q4





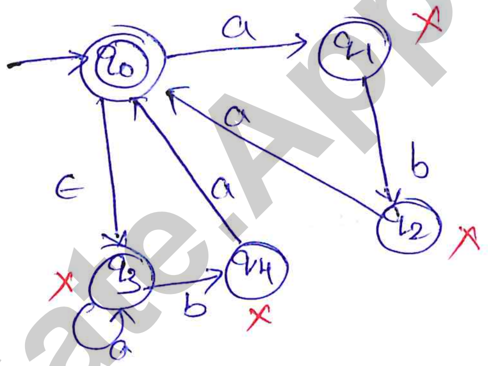


⇒

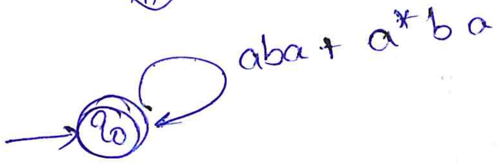


$\Rightarrow \quad ((a+b)^3)^* (a+b)^2$

Q12



⇓

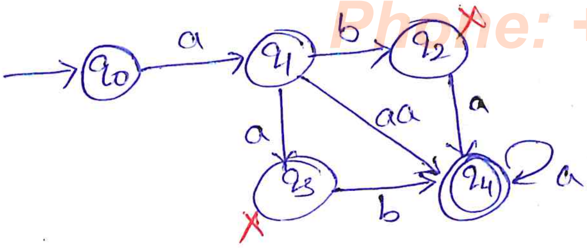


⇓

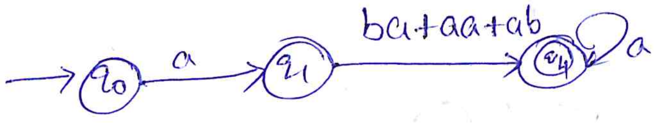
$\boxed{(aba + a^* ba)^*}$

Phone: +91 844-844-0102

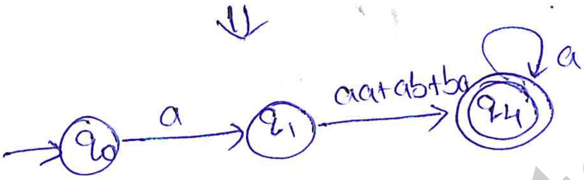
Q13



⇓

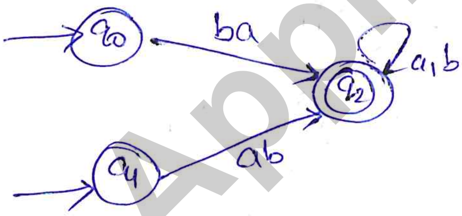


⇓

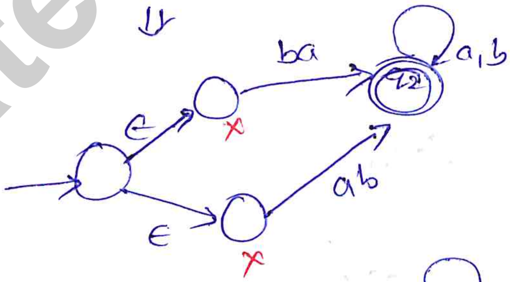


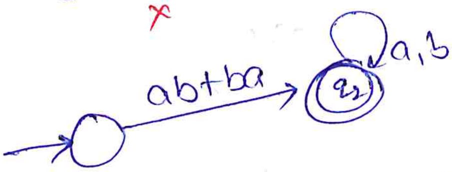
$\boxed{a(aa+ba+ab)a^*}$

Q14



⇓

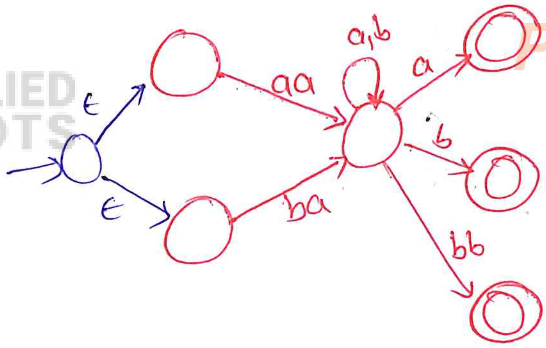




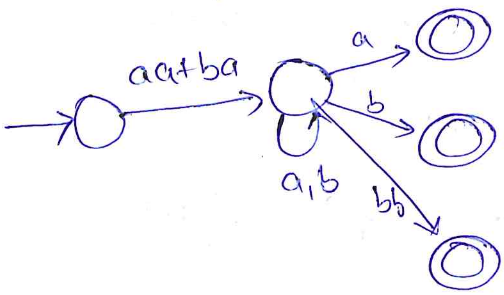
$(ab+ba)(a+b)^*$

Mail: gatecse@appliedroots.com

Q15



⇓



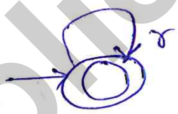
$$(aa+ba)(a+b)^*(a+b+bb)$$

RE → FA

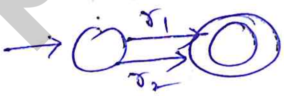
Regular Expression ⟶ Finite Automata

RE → ε NFA
NFA
DFA
min DFA

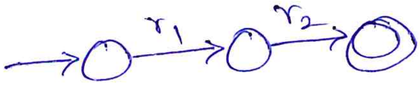
① Kleene closure : r^*



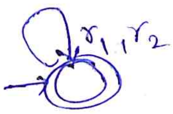
② Union : $r_1 + r_2$



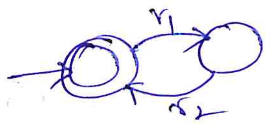
③ Concatenation: $r_1 r_2$



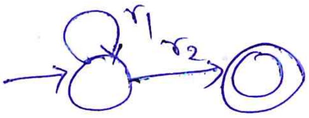
Eg 1: $(r_1 + r_2)^*$



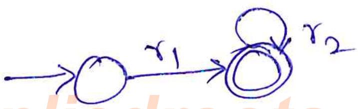
Eg 2:- $(r_1 r_2)^*$



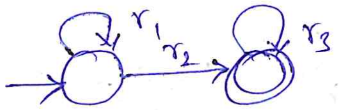
Eg 3: $r_1^* r_2$



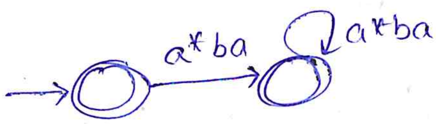
Eg 4: $r_1 r_2^*$

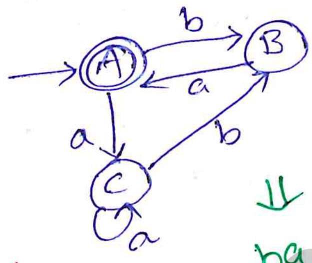


Eg 5: $r_1^* r_2 r_3^*$



Eg 6: $(a^* ba)^*$



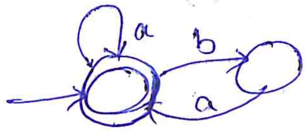


✓

⇓

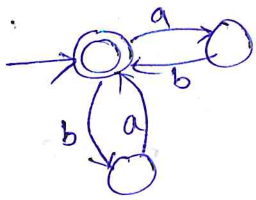
ba, aba, aaba...

✗

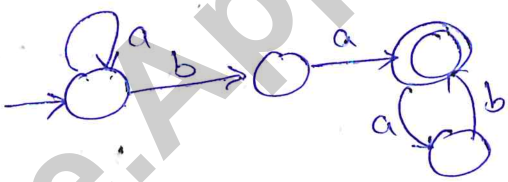


FA accepts a, aa, ... which are not generated by RE

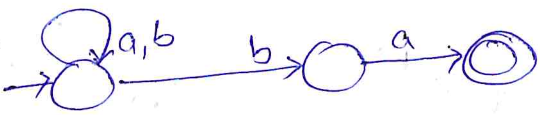
Eg 7: $(ab+ba)^*$



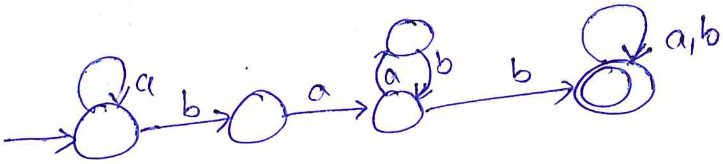
Eg :8 $a^* ba (ab)^*$



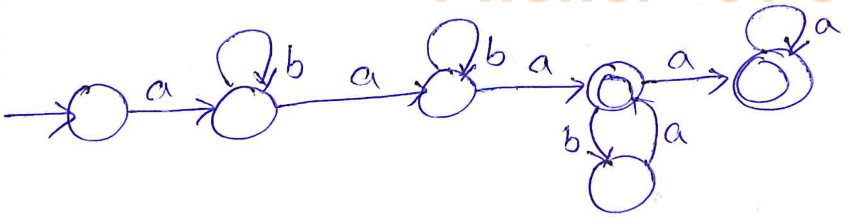
Eg :9 $(a+b)^* ba$



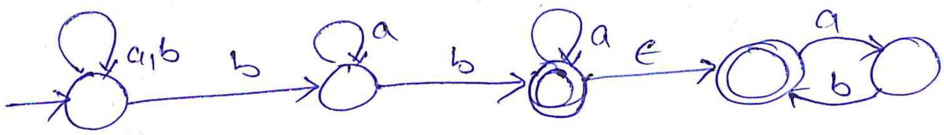
Eg :10 $a^* ba (ab)^* b (a+b)^*$



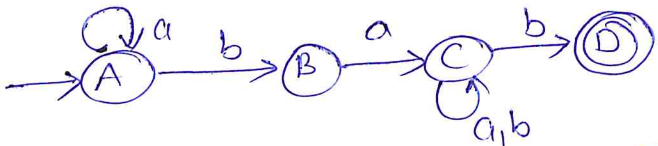
Eg 11) $ab^*ab^*a(ba)^*a^*$ decomposition RE → FA



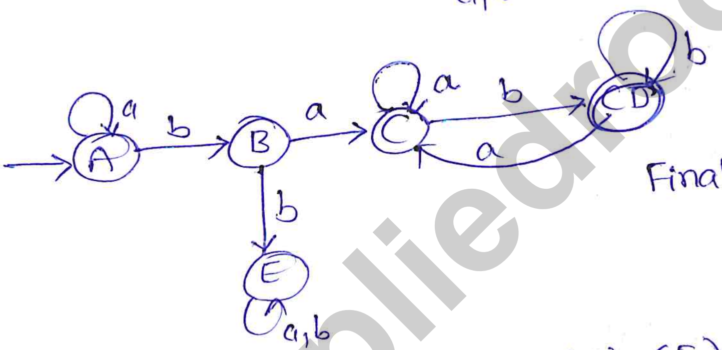
Eg 12: $(a+b)^*ba^*ba^*(ab)^*$



Eg 13: min DFA for $a^*ba(a+b)^*b$



NFA



DFA

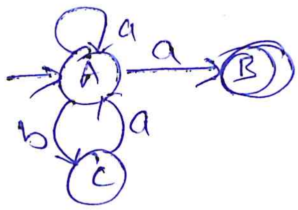
Final state

(ABC) (F) (E)
X

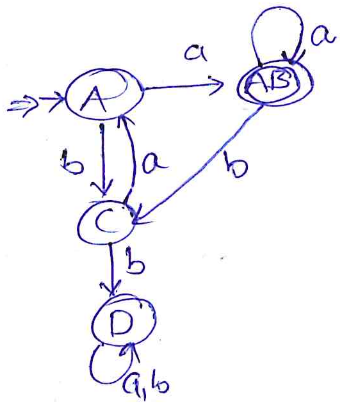
BC AC {A} {B} {C} {F} {E}
X X

Minimal DFA contains 5 states

Eg 14: Min DFA for $(a+ba)^*a$



NFA

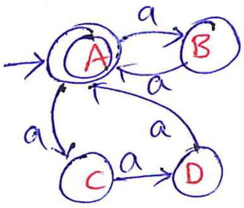


AB is Final State

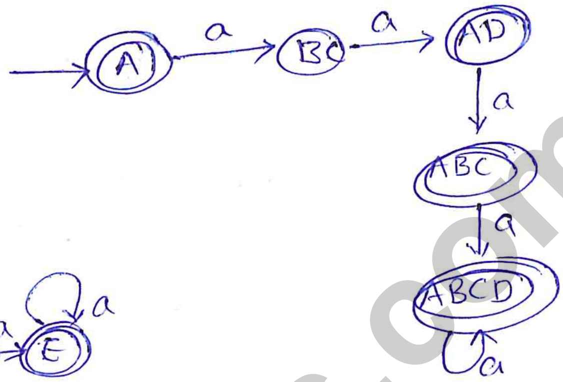
(AC) (D) (B)

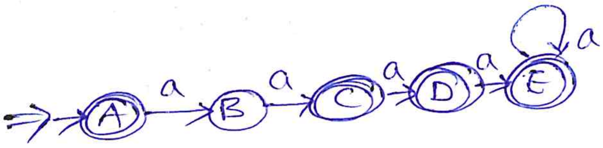
(A) (C) (D) (B)

Eg 15: min DFA for $(aa + aaa)^*$ $\Sigma = \{a\}$

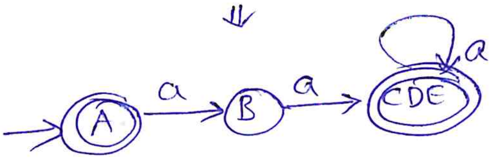


$\Rightarrow$





$\Downarrow$



Min DFA contains 3 states

Algebraic properties of RE operators:

$*$: Kleene closure unary

$+$: union } Binary

$\cdot$: concatenation

DM: Algebraic structures

{ Closure, Associativity, Identity } { SG, monoid, groups, abelian }

① **Closure:** If r_1, r_2 are RE

r_1^* is a RE

$r_1 + r_2$ is a RE

$r_1 \cdot r_2$ is a RE

② **Associativity:** If r_1, r_2, r_3 are RE

$r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$

$r_1(r_2 r_3) = (r_1 r_2) r_3$ ✓

$*$ is a unary operator

③ **Identity:** $r + x = r \Rightarrow x = \phi$ $\quad r \cup \phi = r$ then ϕ is the identity for $+$

$r \cdot x = r$

$\Rightarrow r \cdot x = r$

$x = \epsilon$

$\Rightarrow r \cdot \epsilon = r \Rightarrow \epsilon$ is the identity element for $\cdot$

④ **Annihilator:**

$r + x = x \Rightarrow r \cup x = x$ No $\Rightarrow \nexists$ an ~~anih~~ annihilator for $+$

$r \cdot x = x$

$\Rightarrow x = \phi$

$\{\ldots\}\{\ldots\} = \{\}$

ϕ is the annihilator for $\cdot$ operation

$\{a, aa, ab\} \cdot \{\} = \{\}$

⑤ **Commutative property:**

If $r_1\ r_2$ are RE then

$r_1 + r_2 = r_2 + r_1$ ✓ $\quad a+b = b+a$

$r_1 \cdot r_2 = r_2 \cdot r_1$ ✗

$ab = ba$

⑥ **Distributive property:**

$(r_1 + r_2) \cdot r_3 = r_1 \cdot r_3 + r_2 \cdot r_3$ ✓

Right distributive

$r_1(r_2 + r_3) = r_1 \cdot r_2 + r_1 r_3$ left distributive

$(r_1 \cdot r_2) + r_3 = (r_1 + r_3)(r_2 + r_3)$ ✗

$r_1 = a$

$r_2 = b$

$r_3 = c$

$(ab) + c \neq (a+b)(b+c)$

$r_1 + (r_2 \cdot r_3) \neq (r_1 + r_2)(r_1 + r_3)$ ✗

7. Idempotent property:

$r_1 + r_1 = r_1 \Rightarrow r_1 \cup r_1 = r_1$

$r \cdot r \neq r \Rightarrow$ concatenation does not satisfy the idempotant prop...

Not always

set

Algebraic structure $\rightarrow$ set, op (closure) $\leftarrow (R, *)$

RE ; $+, \cdot, *$

Semi group $\rightarrow$ Associativity

Monoid $\rightarrow$ Identity $(R, +)$ $(R, \cdot)$

(*) **Group** $\rightarrow$ Inverse does not exist for Regular Expression

$\downarrow$

(Real, +) $\rightarrow$ addition

$r + (-r)$

$10 + (-10) = 0$

(*) **Abelian Group** $\rightarrow$ Commutative property

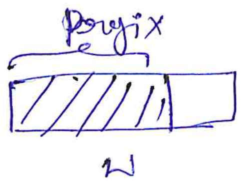
$\Downarrow$

Not exist for RE

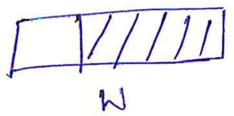
prefix operator & closure:

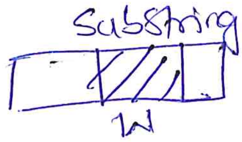
L : RL

$L = \{aba, aab, abab\}$



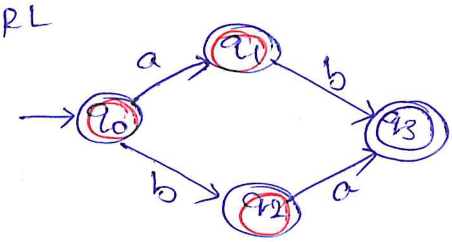
$L' = \{ab, a, aba, aa, aab,$
$ab, aba, abab\}$





If L is a Regular language, then prefix(L) is also regular.

L : RL



$L' = prefix(L)$

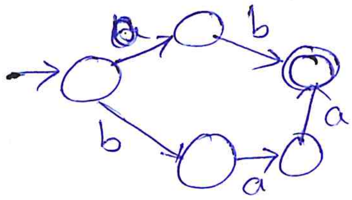
$L = \{ab, ba\} \Rightarrow L' = \{\epsilon, a, ab, b, ba\}$

Reversal operator & closure:

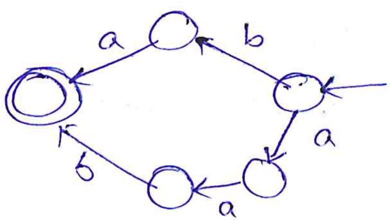
w

If L is a RL then rev(L) is also a RL.

$L = \{ab, baa\} \longrightarrow$ FA



$rev(L) = \{ba, aab\}$



$L = \{ab, abb\}$ Rev(L)

$L^R \rightarrow$

Quotient operator & closure:

let $L_1 = \{010, 101, 110, 001\}$

$L_2 = \{10, 01\}$

$\frac{L_1}{L_2} = \{0, 1\}$

Right cancelation is the default

$\frac{0\cancel{10}}{\cancel{10}}$ $\frac{1\cancel{01}}{\cancel{01}}$

$\frac{1\cancel{10}}{\cancel{10}}$ $\frac{0\cancel{01}}{\cancel{01}}$

$\boxed{\frac{010}{11} = \phi}$

left cancelation

$\frac{\cancel{01}0}{\cancel{01}}$

$\frac{L_1}{L_2} = \{x \mid xy \in L_1 \text{ and } y \in L_2\}$

Right Cancelation

$\frac{L_1}{L_2} = \{x \mid xy \in L_1 \text{ and } x \in L_2\}$

left cancelation.

If $L_1 \& L_2$ are RL then $\frac{L_1}{L_2}$ is also Regular

substitution:

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

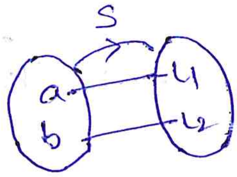
Δ^* = set of all words using Δ

2^{Δ^*} = power set of Δ^*

= set of all Languages using Δ

$\{\{\}, \{\}, \{\} \ldots\}$

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



$L_1 \in 2^{\Delta^*}$

$L_2 \in 2^{\Delta^*}$

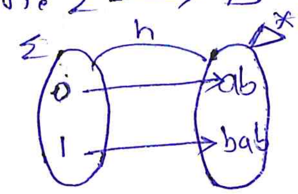
$S : \Sigma \longrightarrow 2^{\Delta^*}$

If L_1 & L_2 are regular on Δ then language obtained after substitution is also regular.

Homomorphism & closure: L is a RL over Σ & $h: \Sigma \to \Delta^*$ then $h(L)$ is also a regular over Δ

Special type of substitution

$h: \Sigma \longrightarrow \Delta^*$ $\quad \Sigma = \{0,1\}$ $\quad \Delta = \{a,b\}$



$h(0) = ab$ $\quad L = \{ab\}$

$h(1) = bab$

$L \xrightarrow{h} h(L)$

$\Downarrow$

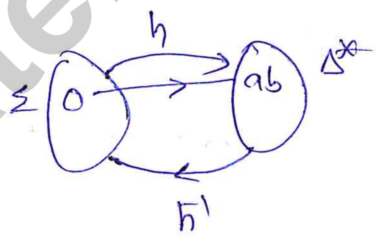
homomorphic image of L

Inverse Homomorphism & closure:

Σ, Δ $\quad h: \Sigma \longrightarrow \Delta^*$

$\uparrow$

L $\quad h^{-1}: \Delta^* \longrightarrow \Sigma$



$h^{-1}(L) = \{ x \in L \mid h(x) \in h(L) \}$

strings $\longrightarrow$ symbols

If L is a RL then $h^{-1}(L)$ is also RL

RL or Non-RL? [pumping lemma for RL]

(Σ)

For any RL, L $\exists$ an integer n $\rightarrow$ is dependent on L

$L \subseteq \Sigma^*$

s.t $\forall z \in L$ and $|z| \geq n$

(i) $z = uvw$

(ii) $|uv| \leq |z|$ then $uv^iw \in L \ \forall i \geq 0$

(iii) $|v| \geq 1$ or $|v| \neq 0$ — u, v, w, z are words built using Σ

Eg 1: $L = \{a^m b^m \mid m \geq 1\}$ (n)

proof by contradiction

✓ let assume L is a RL

$z \in L$

$z = a^k b^k$ choose a k st $|a^k b^k| = 2k \geq n$

u: a^{k-1}, v: a, w: b^k

$|uv| = |a^{k-1}a| = |a^k| = k \leq 2k$

$|v| \neq 0 \Rightarrow |v| = 1$

Hence $uv^iw \in L \ \forall i \geq 0$

for $i = 2$ $a^{k-1} a^2 b^k = a^{k+1} b^k \notin L$

$\Rightarrow$ L is non-Regular

$uv^iw \Rightarrow uv^iw$
uv^1w
uv^2w
uv^3w
$\vdots$

pumping any number of v's to the Finite automata

Eg 2: $L = \{a^m b^p \mid m > p\}$

Let L is a RL

$z \in L$

$z = a^{k+1} b^k$ choose k s.t $|z| \geq n$

$2k+1 \geq n$

u v w

a^{k+1} b b^{k-1}

$|uv| \leq |z|$

$|v| \neq 0$

$uv^iw \in L \ \forall i \geq 0$

$i = 2 \Rightarrow a^{k+1} b^2 b^{k-1}$

$\Rightarrow a^{k+1} b^{k+1} \notin L$

$\Rightarrow$ $\boxed{\text{L is not a Regular}}$

Eg 3: $L = \{a^{3^n} \mid n \geq 1\}$

$z \in L$

u v w

a^{3^n-1} a ϵ

$|uv| \leq |z|$ ✓

$|v| \neq 0$ ✓

$uv^iw \in L \ \forall i \geq 0$

$i = 2$

$a^{3^n-1} a^2 \epsilon$

$\Rightarrow \boxed{a^{3^n+1} \notin L} \Rightarrow$ Not Regular

Special case of pumping lemma when $\Sigma = \{a\}$ Singleton set

↳ Lengths of the string must follow Arithmatic progression for a L to be a RL

uv^iw

$a(aa)^ia \Rightarrow \underbrace{aa}_{2}, \underbrace{aaaa}_{4}, \underbrace{aaaaaa}_{6} \ldots$

Eg1: $L = \{a^{3n} \mid n \geq 1\}$ $\Sigma = \{a\}$

RL

$a^3, a^6, a^9, a^{12} \ldots$

3 3 3.

Eg2: $L = \{a^{2n+1} \mid n \geq 0\}$

1, 3, 5, 7, ... AP

Eg3: $L = \{a^{n^2} \mid n \geq 0\}$ Non-Regular

0, 1, 4, 9 ...

1 3 5

Eg4: $L = \{a^{n^3+n^2+1} \mid n \geq 0\}$ Non-Regular

1, 3, 13, ...

2 10

Eg5: $L = \{a^{2^n} \mid n \geq 0\}$ Non-Regular

$= \{a, \ldots\}$

1, 2, 4, 8, 16, ...

Eg6: $L = \{a^p \mid p \text{ is a prime}\}$

2, 3, 5, 7, 11, 13, 17 .... ≠ AP

Non-Regular

Grammars & Languages:

RE, RL $\longleftrightarrow$ FA

English Grammar: Rules to produce/generate correct Language.

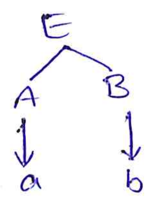
TOC: production rules that generate words in a Language.

G: $E \rightarrow AB$ Variables: E, A, B

$A \rightarrow a$ Terminals: a, b

$B \rightarrow b$ Starting Symbol: E

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



$\boxed{W = ab}$

Language generated by the Grammar. $L(G) = \{ab\}$

$G = (V, T, P, S)$

V: Variables

T: Terminals

P: productions

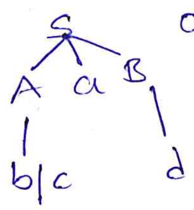
S: Start Symbol

Eg1:

$S \rightarrow AaB$

BNF $A \rightarrow b \mid c \Rightarrow \begin{cases} A \rightarrow b \\ A \rightarrow c \end{cases}$

$B \rightarrow d$



one starting symbol

Derivation Tree

$\Rightarrow$ bad
cad

$L(G) = \{bad, cad\}$

$S \rightarrow \boxed{AaB} \rightarrow \boxed{baB} \rightarrow bad$

$\phantom{S \rightarrow \boxed{AaB}} \rightarrow \boxed{caB} \rightarrow cad$

$\Downarrow$

Sentential forms

(call intermediate steps)

Eg2:

G1: $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$

G2: $S \rightarrow Ab$, $A \rightarrow a$

$L(G_1) = \{ab\}$

$L(G_2) = \{ab\}$

$G_1 = G_2$ iff

$L(G_1) = L(G_2)$

Eg 2:

$S \to ABC$
$A \to a$
$B \to b$
$C \to c$

$S \to ABC$, $A \to a$, $B \to b$, $C \to c$

$\Rightarrow S \to abc$

$L(G) = \{abc\}$

Backus-Naur Form: (BNF)

$A \to \alpha_1$
$A \to \alpha_2$
$A \to \alpha_3$

$\Rightarrow A \to \alpha_1 | \alpha_2 | \alpha_3$ BNF

Eg 1:

$S \to aS$
$S \to b$
$\Downarrow$
$S \to aS | b$

Eg 2:

$S \to AaB$
$A \to aA$, $A \to b$ $\}$ $A \to aA | b$
$B \to d$, $B \to e$ $\}$ $B \to d | e$

Eg 3:

$A \to a$
$A \to b$
$A \to \epsilon$
$\Downarrow$
$A \to a | b | \epsilon$

Recursive production:

$A \to aA$ (Right)
$A \to Aa$ (left)
$A \to aAb$ (General)
$A \to AaA$ (L & R)

$A \to aB$
$B \to bA$
$\Rightarrow A \to abA$ (Indirect Recursion)

Grammar consisting of recursive production only
$\Downarrow$
Recursive Grammar

(P4)

Recursive & Non-Recursive Grammar:

⇓ Recursive production Rules

⇓ Non-Recursive production Rules

Recursive Grammar always generates ∞-Language

Non-Rec Grammar always generates a finite Language.

Eg1: ① $A \to aA \mid b$ $\quad A \to aA \to aaA \to aab$

$\{b, ab, aab, \dots\}$ $\boxed{a^*b}$

② $A \to Aa \mid b$

③ $A \to aA \mid \epsilon$

④ $A \to Aa \mid \epsilon$

⑤ $A \to aA \mid a$

⑥ $A \to Aa \mid a$

⑦ $A \to aA \mid bA \mid \epsilon$

⑧ $A \to aA \mid bA \mid a \mid b$

② $A \to Aa \mid b$

$\{b, ba, baa, baaa, \dots\}$

$A \to Aa \to Aaa \to Aaaa$

$A \to b$, $Aa \to b$, $Aaa \to b$, $Aaaa \to b$

$L(G) = \boxed{ba^*}$

③ $A \to aA \mid \epsilon$

$A \to aA \to aaA \to aaaA \dots$

$A \to \epsilon$, $aA \to a$, $aaA \to aa$, $aaaA \to aaa$

$= \{\epsilon, a, aa, aaa, \dots\} = a^*$

④ $A \rightarrow Aa \mid \epsilon$

$\Rightarrow \boxed{a^*}$

⑤ $A \rightarrow Aa \mid a$

$\Rightarrow \boxed{a^+}$

⑥ $A \rightarrow aA \mid a$

$\Rightarrow \boxed{a^+}$

⑦ $A \rightarrow aA \mid bA \mid \epsilon$

$= \{\epsilon, a, b, ab, bb, ba, bb, \ldots\}$

$\Rightarrow (a+b)^*$

⑧ $A \rightarrow aA \mid bA \mid a \mid b$

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

$\Rightarrow (a+b)^+$

Solved Examples / problems :

Q1 Find Language generated by

$S \rightarrow AB$
$A \rightarrow a \mid b$
$B \rightarrow c$

$S \rightarrow aB$
$\rightarrow ac$

$S \rightarrow bB$
$\rightarrow bc$

$L(G) = \{ac, bc\}$

Q2 Language generated by

$S \rightarrow AaB$
$A \rightarrow b$
$B \rightarrow c \mid d$

$S \rightarrow AaB$, $A \rightarrow b$, $B \rightarrow c$, d

$L(G) = \{bac, bad\}$

Q3

$S \to AB$

$A \to a|b$

$B \to c|d$

$S \to AB$

a or b   c or d

$\Rightarrow \{ac, ad, bc, bd\}$

Q4 $S \to AB|BA$

$A \to a|b$

$B \to c|d$

$S \to AB$

$S \to BA$

$L(G) = \{ac, ad, bc, bd, ca, cd, da, db\}$

Q5 $S \to AaBb$

$A \to a|b|\epsilon$

$B \to b|c$

$S \to AaBb$

$a|b|\epsilon$ $b|c$

$L(G) = \{aabc, babb, aacb, abb, acb\}$

Q6 $S \to Ab$

$A \to bB|b$

$B \to c|d|\epsilon$

$S \to Ab$

b or bB

c   d ϵ

$= \{bb, bcb, bdb\}$

Q7 $S \to aSb|\epsilon$

$L(G) = \{\epsilon, ab, aabb, aaabbb, \ldots\ldots\}$

aSb

aaSbb

aaaSbbb

$\downarrow$

ϵ

$\Rightarrow \{a^n b^n \mid n \geq 0\}$

Q8 $S \to aaSb|\epsilon$

$L(G) = \{\epsilon, a^2b, a^4b^2, a^6b^3, \ldots\}$

$= \{a^{2n}b^n \mid n \geq 0\}$

Q9 $S \rightarrow aSa \mid bSb \mid \epsilon$

$= \{\epsilon, aa, bb, abba, baab \ldots\}$

$= \{ww^R \mid w \in (a+b)^*\}$

$\Downarrow$

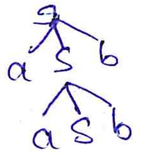
Palindrome

Q10 $S \rightarrow aSb \mid aAb$

$A \rightarrow aA \mid \epsilon \Rightarrow a^*$

$S \rightarrow aSb \mid aa^*b$

$S \rightarrow aSb \mid a^+b$



$a^n S b^n$

$a^n a^+ b b^n$

$a^n a a^* b b^n$

$\Rightarrow a^{n+1} a^* b^{n+1}$

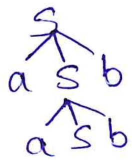
$\Rightarrow \{a^m b^n \mid m \geq n,\ n \geq 1\}$

Q11 $S \rightarrow aSb \mid aBb$

$B \rightarrow bB \mid \epsilon$

$\Rightarrow \boxed{B \rightarrow b^*}$

$S \rightarrow aSb \mid ab^*b$



$a^n S b^n \Rightarrow a^{n+1} b^* b^{n+1}$

$a^n a b^* b b^n \Rightarrow \{a^m b^n \mid n \geq m \geq 1\}$

(108)

Q12 $S \rightarrow aSb \mid aAb$

$A \rightarrow cA \mid c$

$\Rightarrow A \rightarrow c^+$ $\quad S \rightarrow aSb \mid ac^+b$

$a^n S b^n$

$a^n a c^+ b b^n$

$\Rightarrow \{ a^m c^n b^m \mid m \geq 1, n \geq 1 \}$ $\quad a^{n+1} c^+ b^{n+1}$

$a^m c^+ b^m \quad m \geq 1$

Q13 $S \rightarrow aSa \mid bSb \mid c$

$aSa \quad bSb \quad c$

$aca \quad bcb$

aSa

$\Rightarrow$ $abSba$ $\quad L(G) = \{ wcw^R \mid w \in (a+b)^* \}$

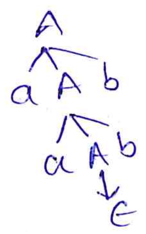
$\downarrow$

$abcba$

Q14 $S \rightarrow AB$

$A \rightarrow aAb \mid \epsilon$ $\quad \{a^n b^n \mid n \geq 0\}$

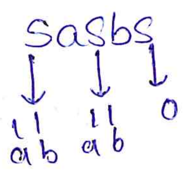
$B \rightarrow cB \mid \epsilon$ $\quad \Rightarrow c^*$



$L = \{ a^n b^n c^m \mid m \geq 0, n \geq 0 \}$

Q15 $S \rightarrow SaSbS \mid SbSaS \mid \epsilon$

$\Rightarrow \{\epsilon, ab, ba, \ldots \}$ $\quad \Rightarrow L(G) = \{ w \in \{a,b\}^* \mid |w_a| = |w_b| \}$



Q16) $S \rightarrow aSb \mid aAb$

$A \rightarrow cAd \mid cd$

$\Rightarrow S \rightarrow aSb \mid ac^m d^m b \quad m \geq 1$

$A \rightarrow cd$

$\rightarrow ccdd$

$\rightarrow cccddd \ldots$

$a^n S b^n$

$a^n\, a c^m d^m b\, b^n$

$a^{n+1} c^m d^m b^{n+1}$

$A \rightarrow c^m d^m \mid m \geq 1$

$\Rightarrow \{ a^k c^m d^m b^k \mid m \geq 1, k \geq 1 \}$

Desiging Grammar for Languages-1

Q1) Construct a grammar that generates

(a) all strings using $\Sigma = \{a, b\}$ including ϵ $\Rightarrow (a+b)^*$

generate()
↳ recursive R
↳ basecase b

$S \rightarrow \underbrace{aS \mid bS}_{R} \mid \underbrace{\epsilon}_{b}$

a ⇝
b ⇝
ϵ

(b) Excluding ϵ

$S \rightarrow \underbrace{aS \mid bS}_{R} \mid \underbrace{a \mid b}_{b}$

Q2) (a) Strings that start with 'b'

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

Σ^*

$S \rightarrow bA$

$A \rightarrow aA \mid bA \mid \epsilon$

(b) Start with 'ab'

$ab \rightarrow \Sigma^*$

$S \rightarrow abA$

$A \rightarrow \epsilon \mid aA \mid bA$

3 (a) Strings that end in 'a' $\Sigma^* a$

$S \rightarrow Aa$

$A \rightarrow \epsilon | aA | bA$

(b) Strings that end in 'ab' $\Sigma^* ab$

$S \rightarrow Aab$

$A \rightarrow \epsilon | aA | bA$

4 (a) Strings that contain 'ab' as Substring

$S \rightarrow AabA$ $\Sigma^* ab \Sigma^*$

$A \rightarrow \epsilon | aA | bA$

5 (a). Strings that start & end in a

$S \rightarrow aAa | a$ $a(a+b)^* a + a$

$A \rightarrow aA | bA | \epsilon$

(b) Starts & ends in same symbol $a \Sigma^* a$

a

$S \rightarrow a | b | aAa | bAb$ $b \Sigma^* b$

b

$A \rightarrow \epsilon | aA | bA$

(c) Starts & ends in different symbols $a \Sigma^* b$

$b \Sigma^* a$

$S \rightarrow aAb | bAa$

$A \rightarrow aA | bA | \epsilon$

6 (a) Third symbol from left is 'b' $\_\ \_\ b\ \Sigma^*$

$S \to AbB$ $\Rightarrow$ $S \to AAbB$

$A \to aa|ab|ba|bb$ $\quad$ $A \to a|b$

$B \to \epsilon|aB|bB$ $\quad$ $B \to \epsilon|aB|bB$

6 (b) 4th symbol from right is 'a'.

$\underset{5}{\underline{\Sigma^*}}\ \underset{4}{\underline{a}}\ \underset{3}{\underline{\ }}\ \underset{2}{\underline{\ }}\ \underset{1}{\underline{\ }}$

A $\quad$ B B B

$S \to AaBBB$

$A \to \epsilon|aA|bA$

$B \to a|b$

7 (a) $\#a's = 2$ $\quad b^*ab^*ab^*$

$S \to BaBaB$

$B \to \epsilon|bB$

(b) $\#a's \le 2$ $\quad b^*(a+\epsilon)b^*(a+\epsilon)b^*$

B $\quad$ A

$S \to BABAB$

$A \to a|\epsilon$

$B \to \epsilon|bB|aB$

(c) $\#a's \ge 2$ $\quad \underset{A}{\underline{\Sigma^*}}\ a\ \Sigma^*\ a\ \Sigma^*$

$S \to AaAaA$

$A \to \epsilon|aA|bA$

(d) $\#a's = \text{even}$ $\quad b^*,\ \overbrace{(b^*ab^*ab^*)^*}^{X}$

0 $\quad$ B $\quad$ 2 $\quad$ B $\quad$ B

A

$S \to B\,|\,BaBaBS$

$B \to \epsilon|bB$

$\boxed{\begin{array}{l} S \to B|X \\ B \to \epsilon|bB \\ X \to \epsilon|AX \\ A \to BaBaB \end{array}}$

112

7 (e) # a's is odd

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

$\underbrace{b^* a b^*}_{1} \; \underbrace{(b^* a b^* a b^*)^*}_{0,2,4,6\cdots}$

let $b^* \leftarrow B$

$X = A^*$

$A = b^* a b^* a b^*$

BaBaB

$$\boxed{\begin{aligned} S &\to BaBX \\ B &\to \epsilon \mid bB \\ X &\to \epsilon \mid AX \\ A &\to BaBaB \end{aligned}}$$

(f) # a's $\equiv 1 \pmod 3$ 1, 4, 7, 10, ...

1 + 0, 3, 6, 9, ...

$\underset{B}{\uparrow}(b^* a b^*) \underbrace{(\underbrace{b^* a b^* a b^* a b^*}_{A})^*}_{X}$

$$\boxed{\begin{aligned} S &\to BaBX \\ X &\to AX \mid \epsilon \\ A &\to BaBaBaB \\ B &\to \epsilon \mid bB \end{aligned}}$$

8 (a) $|W| = 2$

$\overline{A}\ \overline{A}$

$S \to AA$

$A \to a \mid b$

(b) $|W| \le 2$

$\overline{A}\ \overline{A}$

$A \downarrow a \mid b \mid \epsilon$

$S \to AA$

$A \to a \mid b \mid \epsilon$

(c) $|W| \ge 2$

$\overline{A}\ \overline{A}\ \overset{\Sigma^*}{\overline{B}}$

$S \to AAB$

$A \to a \mid b$

$B \to aB \mid bB \mid \epsilon$

8 d) $|w| \equiv 0 \pmod 2$ $0, 2, 4, 6, \ldots$

$[(a+b)^2]^*$

$S \rightarrow \epsilon \mid X$

$X \rightarrow AX \mid \epsilon$

$A \rightarrow BB$

$B \rightarrow a \mid b$

$(\_\ \_)^*$

$a|b \quad a|b$

8 e) $|w| \equiv 1 \pmod 2$

$\underset{A}{\_}\ (\underbrace{\underset{A}{\_}\ \underset{A}{\_}}_{B})^*$

$S \rightarrow AX$

$A \rightarrow a \mid b$

$B \rightarrow AA$

$X \rightarrow \epsilon \mid XB$

8 d) $|w| \equiv 2 \pmod 3$ $2, 5, 8, \ldots \Rightarrow 2 + 0, 3, 6, 9 \ldots$

$(\_\ \_)(\_\ \_\ \_)^*$

$(A\ A)(A\ A\ A)^*$

$S \rightarrow AAX$

$A \rightarrow a \mid b$

$X \rightarrow \epsilon \mid BX$

$B \rightarrow AAA$

9 a) $\{a^m b^n \mid m, n \geq 0\}$

$S \rightarrow AB$

$A \rightarrow \epsilon \mid aA$

$B \rightarrow \epsilon \mid bB$

9 b) $\{a^m b^n \mid m, n \geq 1\}$

$S \rightarrow AB$

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$

9 c $\{a^m b^n \mid m \geq 1 ; n \geq 2\}$

$S \rightarrow AB$

$A \rightarrow a \mid aA$

$B \rightarrow bC$

$C \rightarrow b \mid bC$

$B \rightarrow bC^{+} \Rightarrow \{bb, bbb, \ldots\}$

$\downarrow$

b^{+}

9 d $\{a^m b^n \mid m+n \text{ is even}\}$

m: even | m: odd

n: even | n: odd

$\underbrace{(aa)^*}_{A} \underbrace{(bb)^*}_{B}$ $\quad$ $a\underbrace{(aa)^*}_{A}\underbrace{(bb)^*}_{B}b$

$S \rightarrow XY \mid aXbY$

$X \rightarrow \epsilon \mid AX \rightarrow \boxed{A^*}$

$A \rightarrow aa$

$B \rightarrow bb$

$Y \rightarrow \epsilon \mid BY \rightarrow \boxed{B^*}$

e $\{a^m b^n \mid m+n \text{ is odd}\}$

odd + even $\quad$ even + odd

$\underbrace{(aa)^*}_{A} a \underbrace{(bb)^*}_{B}$ $\quad$ $\underbrace{(aa)^*}_{A} \underbrace{(bb)^*}_{B} b$

$$\boxed{\begin{array}{l} S \rightarrow XaY \mid XbY \\ A \rightarrow aa \\ B \rightarrow bb \\ X \rightarrow \epsilon \mid XA \\ Y \rightarrow \epsilon \mid YB \end{array}}$$

10 a $\{a^m b^n \mid m=n\}$ $\quad \{\epsilon, \underset{S}{ab}, a\underset{S}{ab}b, \ldots\}$

$S \rightarrow \epsilon \mid aSb$

10 b $\{a^m b^n \mid m = 2n\}$

$= \{\epsilon, aab, aaaabb, \ldots\}$

$\boxed{S \rightarrow \epsilon \mid aaSb}$

10 c $\{a^m b^n c^n \mid m, n \geq 1\}$

$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bBc \mid bc$

A $\downarrow$ $a^m \mid m \geq 1$

B $b^n c^n \mid n \geq 1$

10 d $\{a^n b^m c^n \mid m, n \geq 1\}$

$S \rightarrow aSc \mid aBc$

$B \rightarrow b \mid bB$

10 e $\{a^m b^n c^p \mid n = m+p;\ n, m, p \geq 0\}$ $\Sigma = \{a, b, c\}$

$= \{\epsilon, abbc, ab, bc, \ldots\}$
aabbbc

$$\boxed{\begin{array}{l} S \rightarrow AB \\ A \rightarrow \epsilon \mid aAb \\ B \rightarrow \epsilon \mid bBc \end{array}}$$

$a^m b^m \mid m \geq 0$

$b^p c^p \mid p \geq 0$

11 a $L = \{a^m b^n \mid m > n;\ m, n \geq 1\}$

$$\boxed{\begin{array}{l} S \rightarrow aSb \mid aAb \\ A \rightarrow a \mid aA \end{array}}$$

$a^n b^n$

$a\,S\,b$
$\downarrow$
$a, aa, \ldots$

$S \rightarrow aSb$
aaSbb
aaaAbbb

$A \rightarrow a^+ \Rightarrow \boxed{aaaabbb}$

(116)

11 b $\{a^m b^n \mid m < n;\ m, n \geq 1\}$

$$\boxed{\begin{array}{l} S \rightarrow aSb \mid aBb \\ B \rightarrow b \mid bB \end{array}}$$

$S \rightarrow aBb$, $B \rightarrow b^+$

11 c $\{a^m b^n \mid m \neq n,\ m, n \geq 1\}$

$$\boxed{\begin{array}{l} S \rightarrow S_1 \mid S_2 \\ S_1 \rightarrow aS_1b \mid aAb \\ A \rightarrow a \mid aA \\ S_2 \rightarrow aS_2b \mid aBb \\ B \rightarrow b \mid bB \end{array}}$$

12 a $\{wcw^R \mid w, c \in \{a,b\}^*\}$

$$\boxed{\underline{w}\ \underline{c}\ \underline{w^R}, \quad c \rightarrow \Sigma^*}$$

$S \rightarrow aSa \mid bSb \mid aAa \mid bAb \mid \epsilon \mid a \mid b$

$A \rightarrow aA \mid bA \mid \epsilon$

b $\{wcw^R \mid w \in \{a,b\}^*\}$

$S \rightarrow aSa \mid bSb \mid c$

c $\{ww^R \mid w \in \Sigma^*\}$ $\Sigma = \{a,b\}$

$S \rightarrow aSa \mid bSb \mid \epsilon$

13 a $\{w \in \Sigma^* \mid |w_a| = |w_b|\}$ $\Sigma = \{a,b\}$

$S \rightarrow SaSbS \mid SbSaS \mid \epsilon$

13 (b) $\{ w \in \Sigma^* \mid |W_a| = 2|W_b| \}$

$S \rightarrow SaSaSbS \mid SaSbSaS \mid SbSaSaS \mid \epsilon$

S a S a S b S

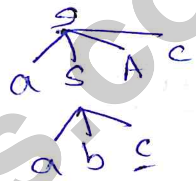
S a S b S a S

S b S a S a S

14 $\{ a^n b^n c^n \mid n \geq 1 \}$

abc, aabbcc, aaabbbccc, ...

$$\boxed{\begin{array}{l} S \rightarrow abc \mid aSAc \quad (1) \\ cA \rightarrow Ac \quad (2) \\ bA \rightarrow bb \quad (3) \end{array}}$$



Best | hardest example

$S \rightarrow aSAc$
$\rightarrow aabcAc$
$\rightarrow aabAcc$
$\rightarrow aabbcc$ ✓

Chomsky hierarchy of Grammars & Languages: "Noam Chomsky"

Type 0 : Recursively Enumerated Grammar : REL → TM : $\boxed{\text{Computer}}$

Type 1 : CSG : CSL ⟷ LBA → syntax & Semantic checks

Type 2 : CFG : CFL ⟷ PushDown Automata → syntax check (CD)
FA + 1 stack

Type 3 : RG : Regular Languages ⟷ FA
↓
pattern Matching

118

Type-0 | Recursively Enumerated | Unrestricted Grammar :

REL
TM

V : Variables

T : Terminals

$\alpha \rightarrow \beta$ $\quad \alpha \in (V+T)^*$

$\beta \in (V+T)^*$

S : Start Symbol

REG

$S \rightarrow aAB$

$aA \rightarrow aB \mid b$

$Ba \rightarrow b$

REL: $L = \{a^{n!} \mid n \geq a\}$

Type-1 | Context-Sensitive Grammar (CSG) $\boxed{CSL \mid LBA}$

$\forall \; \alpha \rightarrow \beta$ $\quad \alpha \in (V+T)^*$

$\beta \in (V+T)^*$

① $|\alpha| \leq |\beta|$

② $\beta \neq \epsilon$

Eg: $S \rightarrow aAB$

$aA \rightarrow aB \mid bb$

$Ba \rightarrow bb$

$\boxed{Ba \rightarrow b}$ ×

Eg: $L = \{a^n b^n c^n \mid n \geq 1\}$

$S \rightarrow abc \mid aSAc$

$cA \rightarrow Ac$

$bA \rightarrow bb$

Type-2 | Context free Grammar (CFG) CFL | PDA

$\alpha \to \beta$

$\Rightarrow \alpha \Rightarrow V$

$\beta \Rightarrow (V+T)^*$ β can be ϵ

Eg: $S \to aSa \mid \epsilon$

$\boxed{bB \to bb}$ ✗

CFL : $\{a^n b^n \mid n \geq 1\}$

Type-3 | Regular Grammar: RG → RL → FA

$A \to xB$

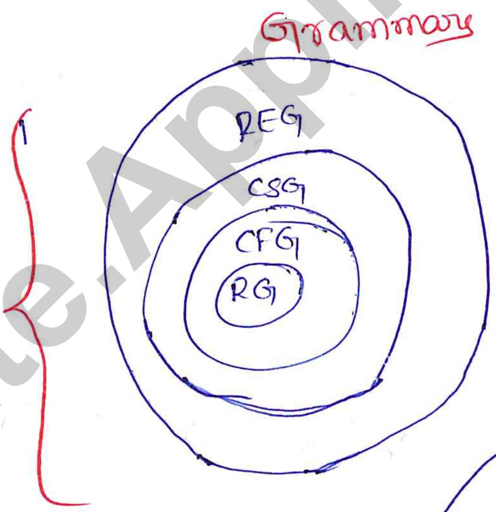
$A \to x$

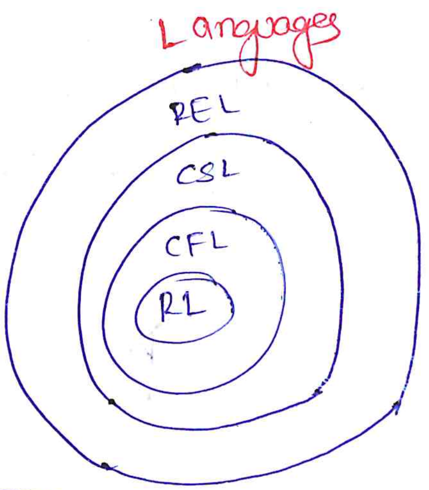
$B \to Bx$

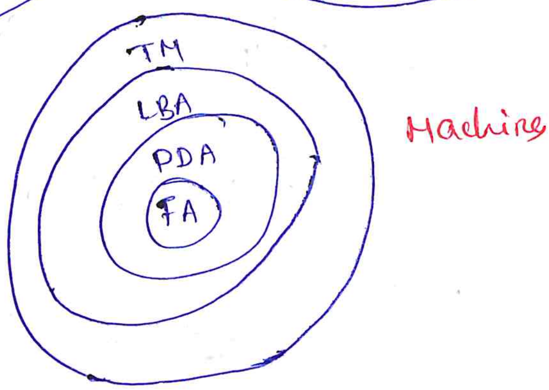
$A, B \in V$

$x \in T^*$ ↳ String of terminals

Eg: $S \to 001S \mid 10 \mid \epsilon$







Regular Grammars & Types:

$A \rightarrow Bx$ — LLG

$A \rightarrow xB$ — RLG

$A \rightarrow x$

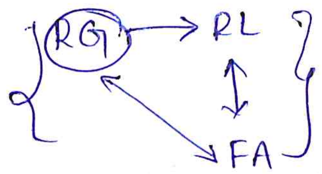
Right Regular Linear Grammar (RLG)

Left Linear Grammar (LLG)
Regular

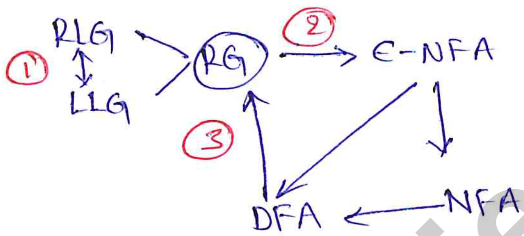
$A, B \in V$

$x \in T^*$

$\boxed{RLG \longleftrightarrow LLG}$



Equivalence of RG & FA:

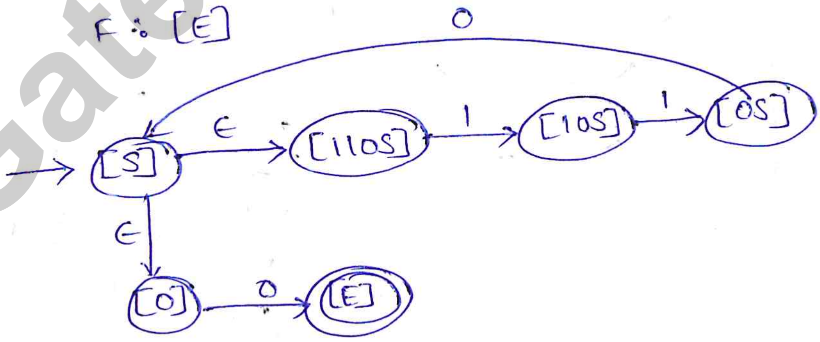


1 RLG to ε-NFA:

$S \rightarrow 110S \mid 0$ $\Sigma = \{0, 1\}$ $Q = \{[S], [110S], [10S], [0S], [0], [\epsilon]\}$
states

q_0: [S]

F: [E]



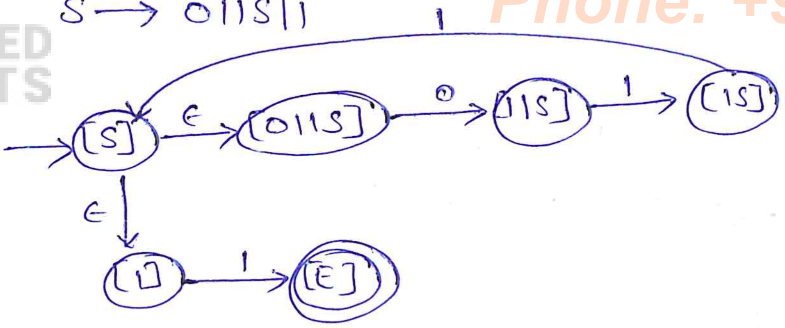
1100

$\rightarrow [S] \xrightarrow{\epsilon} 110S \xrightarrow{1} 10S \xrightarrow{1} 0S \xrightarrow{0} [S] \xrightarrow{\epsilon} [0] \xrightarrow{0} E$

1100 is accepted

$L(G) = \{0, 1100, 110100, \ldots\}$

(2) $S \rightarrow 011S \mid 1$



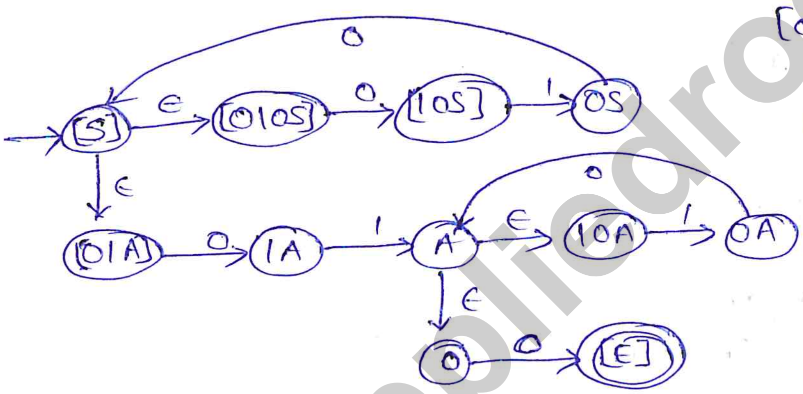
$L(G) = \{1, 0111, 0110111, \dots\}$

$L(M) = \{1, 0111, 0110111 \dots\}$

(3) $S \rightarrow 010S \mid 01A$ $\Sigma = \{0, 1\}$

$A \rightarrow 10A \mid 0$

$Q = \{[S], [010S], [10S], [0S], [01A]\ [1A], [A], [10A], [0A], [E]\}$



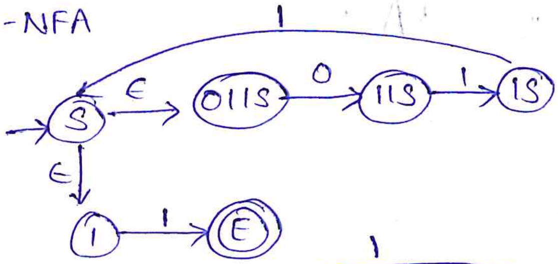
Equivalences : RLG $\longleftrightarrow$ LLG $\longleftrightarrow$ FA

LLG to ε-NFA

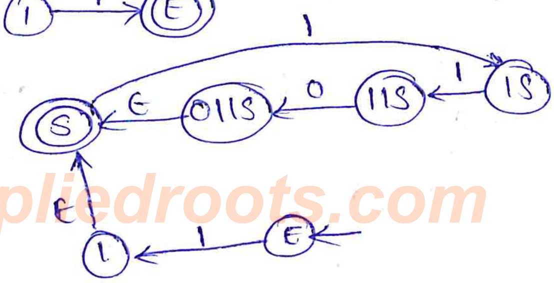
(1) $S \rightarrow S110 \mid 1$

Step(1) : Reverse it $\Rightarrow S \rightarrow 011S \mid 1$ RLG

Step (2) ε-NFA



Step (3) :

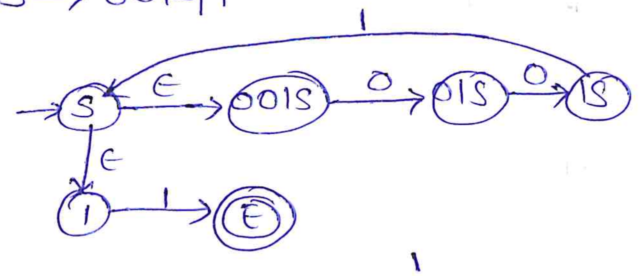


← ε-NFA to LLG

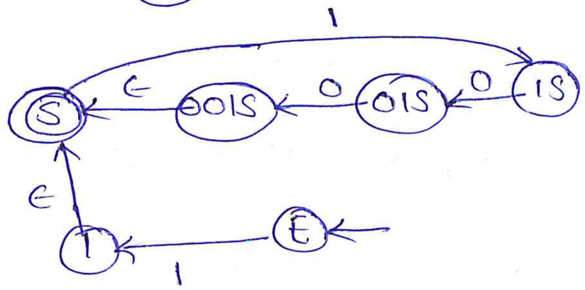
(2) $S \to S100 \mid 1$

(1) $S \to 001S \mid 1$

(2)

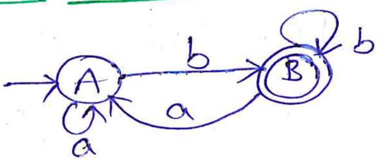


(3)



DFA to RLG:

(1)

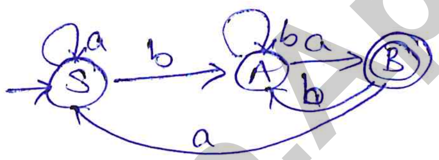


$\Rightarrow A \to aA \mid bB$

$B \to bB \mid aA$

$B \to \epsilon$ (B is the Final state)

(2)



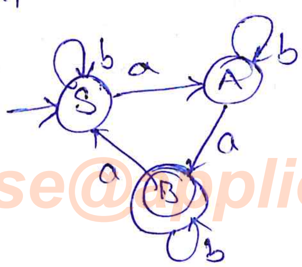
$\Rightarrow S \to aS \mid bA$

$A \to bA \mid aB$

$B \to bA \mid aS$

$B \to \epsilon$

(3) $|w_a| \equiv 2 \mod 3$



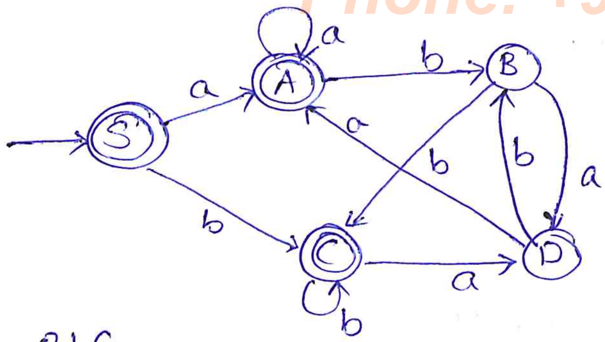
$S \to aA \mid bS$

$A \to aB \mid bA$

$B \to aS \mid bB$

$B \to \epsilon$

(4) Strings that do not ~~contain~~ end in 'ab' or 'ba'

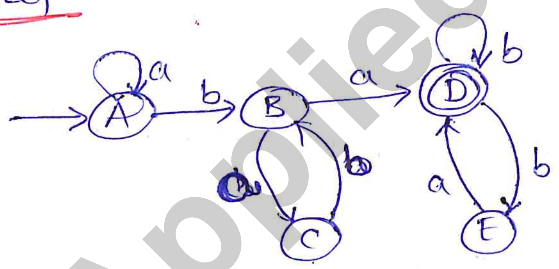


RLG

$$\boxed{\begin{aligned} S &\to aA \mid bC \mid \epsilon \\ A &\to aA \mid bB \mid \epsilon \\ B &\to aD \mid bC \\ C &\to aD \mid bC \mid \epsilon \\ D &\to aA \mid bB \end{aligned}}$$

FA $\longrightarrow$ LLG

(1)

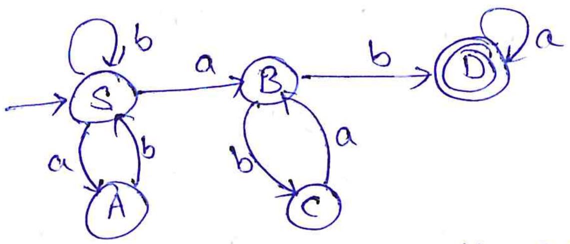


(1) RE: $a^*b\,(ab)^*\,a\,(b+ba)^*$

(2) Reverse RE:

$(ab+b)^*a\,(ba)^*\,b\,a^*$

(3) Draw the finite automata for the Reversed RE



(4) Obtain RLG from the FA

$S \to bS \mid aB \mid aA$

$A \rightarrow bS$

$B \rightarrow bC \mid bD$

$C \rightarrow aB$

$D \rightarrow aD \mid \epsilon$

5. Reverse RLG to get LLG

$S \rightarrow Ba \mid Sb \mid Aa$

$A \rightarrow Sb$

$B \rightarrow Cb \mid Db$

$C \rightarrow Ba$

$D \rightarrow Da \mid \epsilon$

$\boxed{\text{DFA} \rightarrow \text{LLG (double Reversion)}}$

RLG to LLG:

1. $S \rightarrow aA \mid baB \Rightarrow a(ba)^* + ba(aab)^*b$

$A \rightarrow baA \mid \epsilon \Rightarrow (ba)^*$

$B \rightarrow aabB \mid b \Rightarrow (aab)^*b$

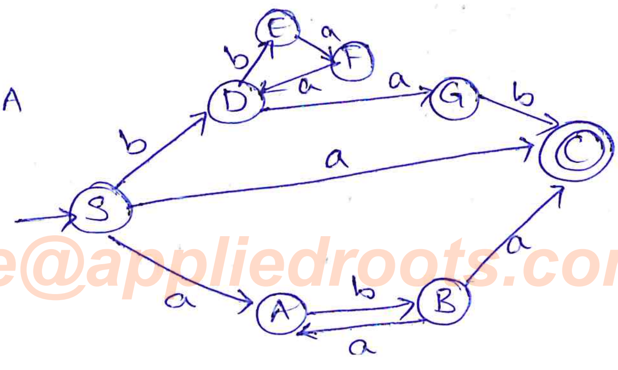
a. RE:

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

b. Reverse

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

c. FA



(d) Generate the RLG

$S \to aA \mid bD \mid aC$

$A \to bB$

$B \to aA \mid aC$

$C \to \epsilon$

$D \to bE \mid aG$

$E \to aF$

$F \to aD$

$G \to bC$

(e) Reverse RLG to obtain LLG

$S \to Aa \mid Db \mid Ca$

$A \to Bb$

$B \to Aa \mid Ca$

$C \to \epsilon$

$D \to Eb \mid Ga$

$E \to Fa$

$F \to Da$

$G \to Cb$

LLG to RLG :-

(1) $A \to A10 \mid B110 \mid 101$

$B \to B011 \mid 01$

(a) Reverse the RHS of production Rules

$A \to 01A \mid 011B \mid 101$ $\quad ((01)^*(101 + 011(110)^*10 + 011B))$

$B \to 110B \mid 10$ $\quad (110)^*10$

RE for A is $(01)^*[101 + 011(110)^*10] + 011(110)^*10$

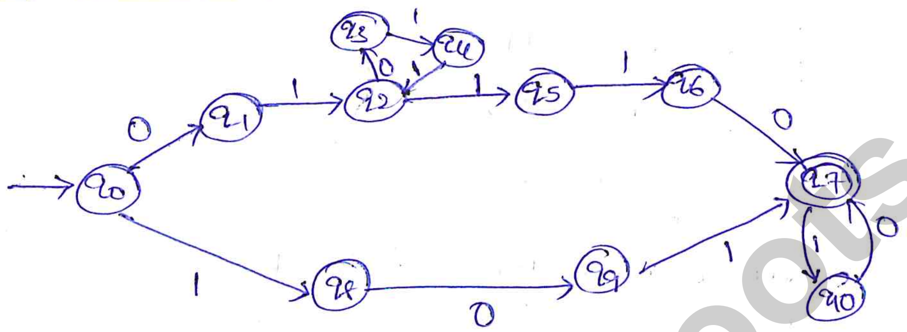
$$(01)^* (101 + 011(110)^* 10) + 011(110)^* 10$$

$$\Rightarrow (01)^* (101 + 011(110)^* 10)$$

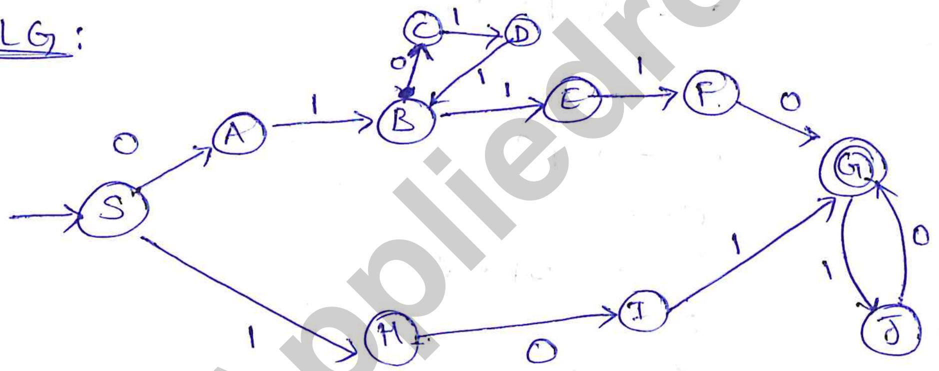
(c) Reverse the RE:

$$(01(011)^* 110 + 101)(10)^*$$

(d) Finite Automata for the RE:



(e) RLG:



(f) $S \rightarrow 0A \mid 1H$ $\Rightarrow$ ~~Reverse to obtain~~

$A \rightarrow 1B$

$B \rightarrow 0C \mid 1E$

$C \rightarrow 1D$

$D \rightarrow 1B$

$E \rightarrow 1F$

$F \rightarrow 0G$

$G \rightarrow 1J \mid \epsilon$

$H \rightarrow 0I$

$I \rightarrow 1G$

$J \rightarrow 0G$

CFG Introduction and optimization:

Context Free Grammar (CFG), CFL, PDA

$A \to \alpha$, $A \in V$

$\alpha \in (V+T)^*$

Single var

Eg: $S \to aSb$, $S \to \epsilon$ — CFG, Not RG

$S \to aS$, $S \to Sa$ — RG

CFL : $L(G) = \{a^n b^n \mid n \geq 0\}$

L_1 : $\{a^m b^n \mid m < n\}$

L_2 : $\{a^m b^n \mid m \neq n\}$

optimization: Elimination of useless symbols / var & Rules

useless symbols

(1) $S \to aA$, $A \to b$, $B \to b$ $\Rightarrow$ $S \to aA$, $A \to b$

B is not derivable from S.

(2) $S \to aA \mid bB$ ✗, $A \to b$ $\Rightarrow$ $S \to aA$, $A \to b$

B does not derive a terminal symbol

(3) $S \to aAb \mid \boxed{Ba}$, $A \to \boxed{aB} \mid ba$, $B \to \boxed{aBC}$ ✗ $\Rightarrow$ $S \to aAb$, $A \to ba$

B and C are useless var

(4) $S \to aAB \mid BC$ ✗, $A \to aB \mid b$, $B \to bA \mid \cancel{BC}$

$B \to bA \Rightarrow bb$ (indirectly derives bb), $A \to b$

$S \to aAB$, $A \to aB \mid b$, $B \to bA$

(5)

$S \to aAB \mid \cancel{BaC} \mid \cancel{BC}$

$A \to aA \mid Ba \mid \cancel{AC} \mid a$

$B \to \cancel{BAC} \mid BB \mid b$

C is a useless var

$S \to aAB$

$A \to aA \mid Ba \mid a$

$B \to BB \mid b$

(6) Unit productions:

$A \to B$, $A, B \in V$ {Simplify}

(1) $S \to aA$, $A \to B \mid b$, $B \to a$

$S \to aA$, $A \to a \mid b$, $\boxed{B \to a}$ useless

$S \to aA$, $A \to a \mid b$

(2) $S \to AaB$, $A \to B \mid aB \mid a$, $B \to C \mid bB \mid b$

$\Rightarrow$ $\cancel{A \to a+}$ $S \to AaB$, $A \to B \mid aB \mid a$, $B \to bB \mid b$

$\Rightarrow$ $S \to AaB$, $A \to aB \mid a \mid bB \mid b$, $B \to bB \mid b$

(3) $S \to ABa \mid BC$ ✗, $A \to aA \mid B \mid a$, $B \to D \mid b$, $D \to b \mid C$

$A \to B$, $B \to D$, $D \to C$ ✗

$S \to ABa$, $A \to aA \mid a \mid b$, $B \to b$, $\boxed{D \to b \text{ ✗}}$

ε-productions:

$A \to \epsilon$; $A \in V$

① $S \to ABC$
$A \to a$
$B \to \epsilon$
$C \to b$

$\Rightarrow$ $S \to ABC$
$A \to a$
$C \to b$

② $S \to AaB$
$A \to b \mid \epsilon$
$B \to c \mid \epsilon$
↓
terminal

$S \to AaB \mid aB \mid Aa \mid a$
$A \to b$
$B \to c$
} No ϵ-productions

③ $S \to bAB \mid BaA$
$A \to aA \mid \epsilon \mid \cancel{AC}$
$B \to bB \mid \epsilon \mid \cancel{BC}$

$S \to bAB \mid BaA \mid bB \mid aA \mid b \mid a \mid Ba \mid bA$
$A \to aA \mid a$
$B \to bB \mid b$

④ $S \to AB$
$A \to a \mid \epsilon$
$B \to b \mid \epsilon$

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

Order of optimizations:

① Remove ϵ-productions
② Remove unit productions
③ Remove useless symbols.

Normal Forms:

① Chomsky Normal Form (CNF)
② Greibach Normal Form (GNF)

Standard forms $\longrightarrow$ Grammar does not generate ϵ

Chomsky Normal Form:

$A \rightarrow BC \qquad A, B, C \in V$

$A \rightarrow a \qquad a \in T$

Eg1: $S \rightarrow aSb \mid ab$

$A \rightarrow a$
$B \rightarrow b$

$S \rightarrow AB \mid \underline{ASB}$
$A \rightarrow a$
$B \rightarrow b$

$$\Rightarrow \boxed{\begin{array}{l} S \rightarrow AB \mid CB \\ C \rightarrow AS \\ A \rightarrow a \\ B \rightarrow b \end{array}} \quad \text{CNF}$$

Eg2: $S \rightarrow aSa \mid bSb \mid \epsilon$

$$\boxed{S' \rightarrow S \mid \epsilon}$$

$$\Rightarrow \boxed{\begin{array}{l} S \rightarrow \epsilon \\ S \rightarrow DA \mid CB \\ D \rightarrow AS \\ C \rightarrow BS \\ A \rightarrow a \\ B \rightarrow b \end{array}}$$

Eg3: $S \rightarrow aAb \mid aB \mid abA$
$A \rightarrow a$
$B \rightarrow b$

$$\Rightarrow \left.\begin{array}{l} S \rightarrow AC \\ S \rightarrow AB \\ S \rightarrow CA \\ C \rightarrow AB \\ A \rightarrow a \\ B \rightarrow b \end{array}\right\} \text{CNF}$$

Eg 4: $S \to CB \mid AB$

$C \to AS$

$B \to b$

$A \to a$

let $w_1 = ab$ $|w_1| = 2$ $w_2 = \overset{4}{\overleftrightarrow{aabb}}$

$S \to AB$
$\to aB$
$\to ab$ } 3

$S \to CB$
$\to ASB$
$\to AABB$
$\to aABB$
$\to aaBB$
$\to aabB$
$\to aabb$ } 7

String of len n $\xrightarrow{CNF}$ $2n-1$ derivations

GNF (Greibach Normal Form):-

$A \to a\alpha$ $A \in V, a \in T, \alpha \in V^*$

↳ String of variables

Eg 1:

$S \to aSb \mid ab$

⇓

$S \to aSB \mid aB$
$B \to b$

Eg 2: $S \to aSa \mid bSb \mid aa \mid bb$

⇓

$S \to aSA \mid bSB \mid aA \mid bB$

$B \to b$

$A \to a$

Eg 3: $S \to aAb \mid bBa$

$A \to bAa \mid b$

$B \to bBa \mid a$

$S \to aAY \mid bBX$

$A \to bAX \mid b$

$B \to bBX \mid a$

$X \to a$

$Y \to b$

Eg 4:

$S \to Aa \mid bBa$

$A \to bB \mid a$

$B \to bBa \mid b$

$S \to AX \mid bBX$
$A \to bB \mid a$
$B \to bBa \mid b$
$X \to a$

$\Rightarrow$ $S \to AX$
$A \to bB \mid a$
$\Downarrow$
$S \to aX \mid bBX$

$S \to aX \mid bBX$
$\boxed{A \to bB \mid a}$
$B \to bBX \mid b$
$X \to a$

$\Rightarrow$ $\boxed{\begin{array}{l} S \to aX \mid bBX \\ B \to bBX \mid b \\ X \to a \end{array}}$

Eg 5:

$S \to aSb \mid ab$

$\Downarrow$

$S \to aSB \mid aB$

$B \to b$

$W_1 = ab$ $|W_1| = 2$

$\left.\begin{array}{l} S \to aB \\ \quad \to ab \end{array}\right\} 2$

$W_2 = aabb$ $|W_2| = 4$

$\left.\begin{array}{l} S \to aSB \\ \quad \to aaSBB \\ \quad \to aabB \\ \quad \to aabb \end{array}\right\} 4$

String of Length n $\xrightarrow{GNF}$ n productions to generate the String.

Decision properties of CFG:

Decidability — TM {Halting problem}

Given a problem, if we can write a Computer program then the given problem is decidable.

① Emptiness & Non-emptiness:

G1: $S \to aA \mid aB$
$A \to aB \mid b$

G2: $S \to AB \mid aB$
$A \to a \mid b$
$B \to AB$

Reduced G:

$S \to aA$ B is useless
$A \to b$

Non-Empty

G2:
$S \to \cancel{AB} \mid \cancel{aB}$ B is useless
$\boxed{A \to a \mid b}$ ✓
$B \to AB$ ✗

Reduced Grammar : Empty Language.

$S \to \square$

$\boxed{\text{Decidable property of CFL}}$

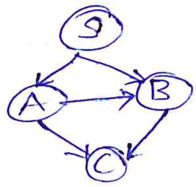
② Finiteness / Infiniteness:

Does the G generate finite algo < Y / N

∃ algorithm

Variable dependency Graph (VDG)

$S \to AB$
$A \to BC \mid a$
$B \to CC \mid b$
$C \to d$



VDG(G)
directed Graph

Loops/Cycles

{ No loops/Cycles ⟶ Finite
{ Loops / Cycles ⟶ Infinite Language

$\boxed{\text{Decidable property}}$

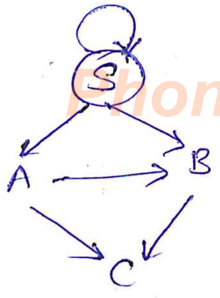
(b)

$S \to SS \mid AB$

$A \to BC \mid a$

$B \to CC \mid a$

$C \to d$



VDG

$\exists$ loop $\Rightarrow$ Infinite Language

Non-Empty Language

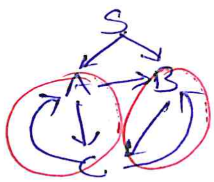
(c)

$S \to AB$

$A \to BC \mid a$

$B \to CC \mid b$

$C \to AB$



Non-Empty

$\exists$ Cycle $\Rightarrow$ Infinite Language.

Membership: $w \in L(G)$? < Y / N

↳ Decidable property

G has to be in CNF

CYK → Coke, Young, Kasami

Dynamic programming

(1) $S \to AB$

$A \to BA \mid SA \mid a$

$B \to BB \mid BS \mid b$

} CNF

(a) $w_1 = ab$

a	b
A	B
S	

Start Symbol

(b) $w_2 = ba$

b	a
B	A
A	

Not Start Symbol

$w_2 \notin L(G)$

(c)

(c) $w_3 = aba$

a	b	a
A	B	A
S	A	
A		

→ Not the Start State

$w = aba \notin L(G)$

① $S \rightarrow AB$

$A \rightarrow BA \mid SA \mid a$

$B \rightarrow BB \mid BS \mid b$

(d) W = bba

b	b	a
B	B	A
B	A	
A		

$\not\ni S$

$W \notin L(G)$

(e) w = abb

a	b	b
A	B	B
A	B	
S		

$abb \in L(G)$

(f) W = aab

a	a	b
A	A	B
A	S	
$\emptyset$		

$\not\ni S$

$W \notin L(G)$

Eg 2:

$S \rightarrow AB \mid AA$

$A \rightarrow BA \mid SA \mid a \mid b$

$B \rightarrow BB \mid BS \mid a$

(a) W = abba

a	b	b	a
A,B	A	A	A,B
S,A	A	A,B	
A,S	A,B		
S,A			

$\rightarrow$ contains S

$abba \in L(G)$

(b)

a	b	a	a
A,B	A	A,B	A,B
S,A	A,B	A,B	
A,S	A,B		
S,A			

$\exists S$

$abaa \in L(G)$

Membership of CFL is decidable

Equality of two Context free Grammar : undecidable property

CYK Algorithm:

(1) Grammar must be CNF

(2) Construct a Triangular table

→ Each row corresponding to one length of substrings.

- Bottom Row – Strings of length 1
- Second Row from Bottom Row – Strings of length 2

⋮

Top Row – String w

$X_{i,i}$ is the set of variables such that $A \rightarrow w_i$ is a production of G

Compare at most n pairs of previously computed sets.

$(X_{i,i}, X_{i+1,j})\ (X_{i,i+1}, X_{i+2,j}) \cdots (X_{i,j-1}, X_{j,j})$

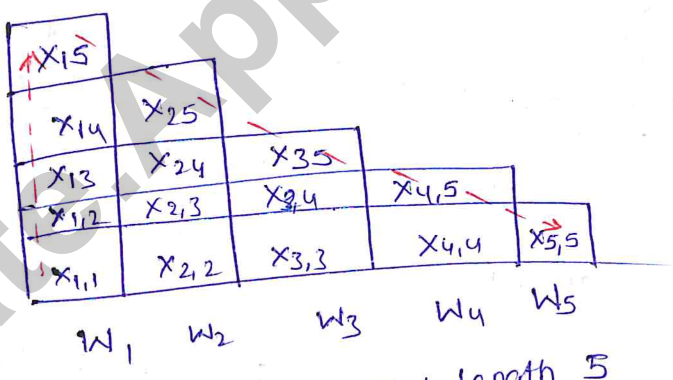


Table for string w of length 5

(1) G: $S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

$\omega = baaba \in L(G)$?

{S,A,C} X_{15}				
ϕ X_{14}	{S,A,C} X_{25}			
ϕ X_{13}	B X_{24}	B X_{35}		
{A,S} X_{12}	{B} X_{23}	{S,C} X_{34}	{S,A} X_{45}	
{B} X_{11}	{A,C} X_{22}	{A,C} X_{33}	{B} X_{44}	{A,C} X_{55}
b	a	a	b	a

$S \to AB \mid BC$
$A \to BA \mid a$
$B \to CC \mid b$
$C \to AB \mid a$

$\boxed{baaba \in L(G)}$

$X_{12} = X_{11} X_{22}$
$B\{A,C\} = BA \ \ BC$

$X_{23} = X_{22} X_{33}$
$\{A,C\}\{A,C\} = \{AA, AC, CA, \underset{B}{\underline{CC}}\}$

$X_{34} = X_{33} X_{44} = \{A,C\}\{B\} = \{\underset{S,C}{\underline{AB}}, CB\}$

$X_{45} = X_{44} X_{55} = \{B\}\{A,C\} = \{\underset{A,S}{BA}, BC\}$

$X_{13} = X_{11} X_{23} \cup X_{12} X_{33}$
$= \{A,S\}\{A,C\}\{B\}\{B\}$
$= \{AA, AC, SA, SC\} \cup \{BB\}$
$= \phi$

$X_{14} = X_{11} X_{24} + X_{12} X_{34} + X_{13} X_{44}$
$= X_{11} X_{24} \cup X_{12} X_{34} \cup X_{13} X_{44}$
$= BB \cup \{A,S\}\{S,C\} \cup \phi$
$= \phi \cup \{AS, AC, BS, SC\} \cup \phi$
$= \phi$

$X_{24} = X_{22} X_{34} + X_{23} X_{44}$
$= \{A,C\}\{S,C\} \cup$
$= \{AS, AC, CS, CC\} \cup \{B\}\{B\}$
$= \{AS, AC, CS, CC\} \cup \underline{BB}$
$B \to CC$

$X_{35} = X_{33} X_{45} \cup X_{34} X_{55}$
$= \{A,C\}\{S,A\} \cup \{S,C\}\{A,C\}$
$= \{\underset{\phi}{\underline{AS, AA, CS, CA}}\} \cup \{SA, SC, CA, \underset{B}{\underline{CC}}\}$
$= B$

$$X_{25} = X_{22}X_{35} + X_{23}X_{45} + X_{24}X_{55}$$

$$= \{A,C\}\{B\} \cup \{B\}\{S,A\} + \{B\}\{A,C\}$$

$$= \{AB, CB, BS, BA, BA, BC\}$$

$$= \{C, A, S\}$$

$$X_{15} = X_{11}X_{25} + X_{12}X_{35} + X_{13}X_{45} + X_{14}X_{55}$$

$$= B\{S,A,C\} + \{A,S\}B + \phi + \phi$$

$$= \{BS, BA, BC, AB, SB\}$$

$$= \{S, A, C\}$$

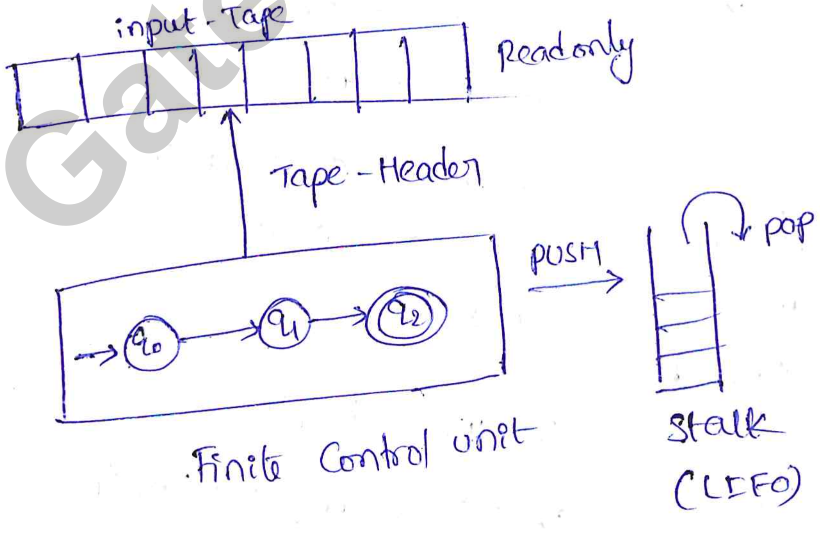
$$\boxed{X_{i,j} = X_{i,i}X_{i+1,j} + X_{i,i+1}X_{i+2,j} \cdots X_{i,j-1}X_{j,j}}$$

Pushdown Automata:

$$\left\{ \begin{array}{c} FA \leftrightarrow RL \leftrightarrow RG \\ \updownarrow \\ RE \end{array} \right\}$$

$$\begin{array}{c} CFG \leftrightarrow CFL \\ \updownarrow \\ PDA \end{array}$$

$$M \leftrightarrow L \leftrightarrow G$$

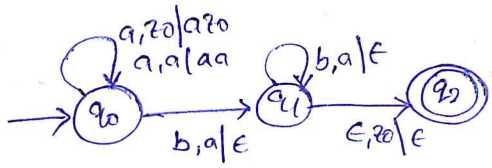


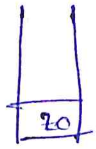
Restrictive Memory

PDA = FA + Stack

$L = \{a^n b^n \mid n \geq 1\}$

Eg 1: $L = \{a^n b^n \mid n \geq 1\}$





Stack

$Q = \{q_0, q_1, q_2\}$

z_0 : Initial Stack top

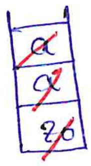
q_0 : Initial State

F : Set of final States

→ Let w = aabb

$\delta(q_0, a, z_0) \longrightarrow q_0, a z_0$

$\delta(q_0, a, a) \longrightarrow q_0, aa$



$\delta(q_0, b, a) \longrightarrow q_1, \epsilon$

Stack is Empty

$\delta(q_1, b, a) \longrightarrow q_1, \epsilon$

$\delta(q_1, \epsilon, z_0) = \underline{q_2}, \epsilon$

Final State

Mathematical Model of PDA:

Q : set of States

q_0 : initial State

F : set of final States

z_0 : initial Stack Top

Γ : Set of Stack Symbols

Σ : Set of input Symbols.

PDA

DPDA $\neq$ NPDA by default

less powerfull

DFA $\approx$ NFA

$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \longrightarrow 2^{Q \times \Gamma^*}$

Current State — Input Symbol — Current Top of Stack — Next State

NPDA ⊃ DPDA

Acceptance
↳ Final State
↳ Empty Stack

Non-deterministic PDA vs Deterministic PDA:

NPDA $\neq$ DPDA

DPDA: $Q \times \Sigma \cup \{\epsilon\} \times \Gamma \longrightarrow Q \times \Gamma^* \longrightarrow$ New Stack Symbol

Current State — input — Current Top — Next State

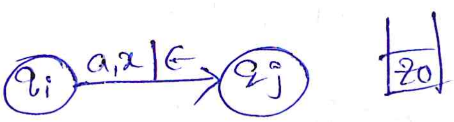
ϵ-transitions are allowed.

NPDA: $Q \times \{\Sigma \cup \{\epsilon\}\} \times \Gamma \longrightarrow 2^{Q \times \Gamma^*}$

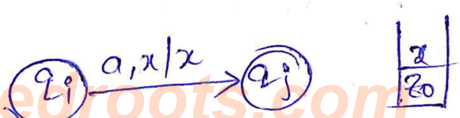
Stack operations in PDA:

① PUSH: $\delta(q_i, a, z_0) = (q_j, xz_0)$

② POP: $\delta(q_i, a, x) = (q_j, \epsilon)$



③ Skip:

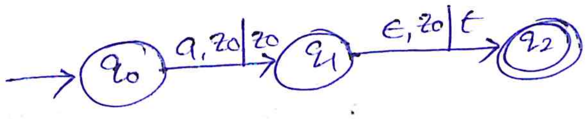


$\delta(q_i, a, x) = (q_j, x)$

Example of PDA:

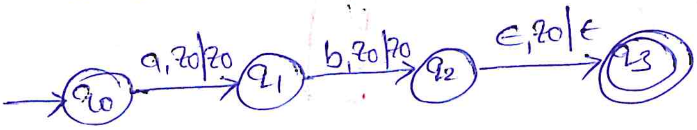
DPDA $\longleftrightarrow$ DCFL NPDA $\longleftrightarrow$ CFL

① $L = \{a\}$



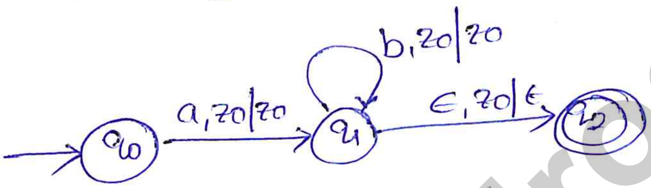
DCFL

② $L = \{ab\}$



DCFL

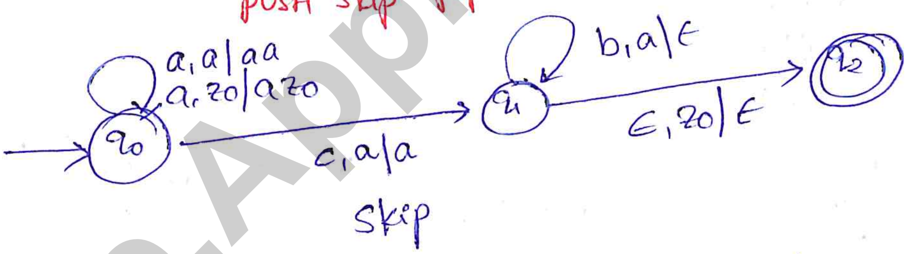
③ $L = \{ab^*\}$



DCFL

④ $L = \{a^n c b^n \mid n \geq 1\}$

PUSH Skip POP



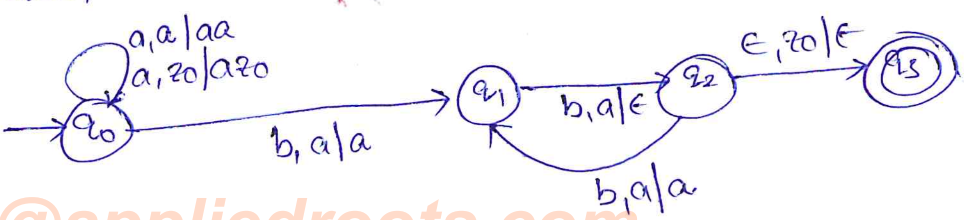
DPDA
⇓
DCFL

DFA: All transitions must be defined.

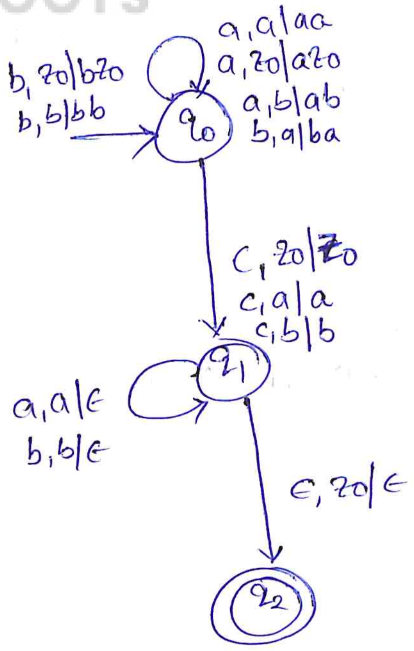
DPDA: Need ~~to~~ not be defined.

⑤ $L = \{a^n b^{2n} \mid n \geq 1\}$

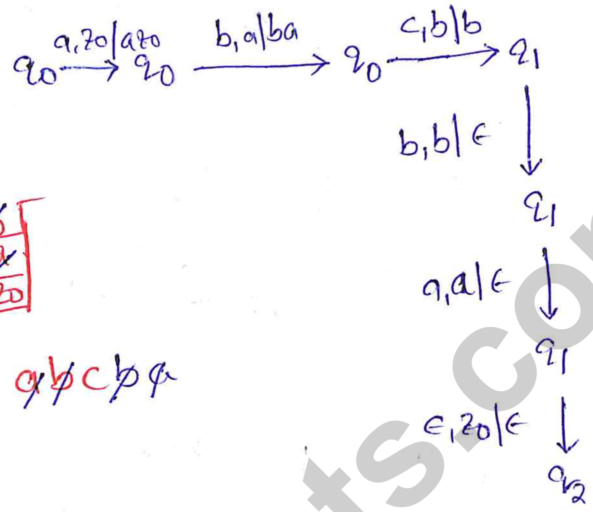
abb, aabbbb, ...

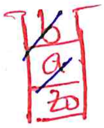


(6) $L = \{ \underset{\text{Push}}{\underline{w}} c \underset{\text{pop}}{w^R} \mid w \in \{a,b\}^* \}$



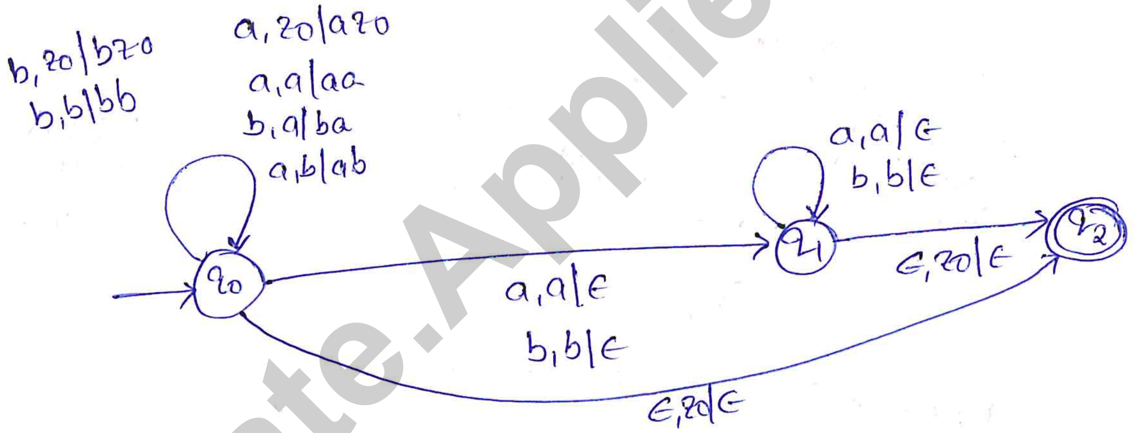
let W = abcba



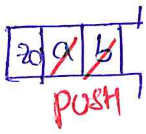


~~a~~ ~~b~~ c ~~b~~ ~~a~~

(7) $L = \{ w w^R \mid w \in \{a,b\}^* \}$ — NPDA



let $w = \underset{\text{push}}{\underline{ab}}\ \underset{\text{pop}}{\underline{ba}}$



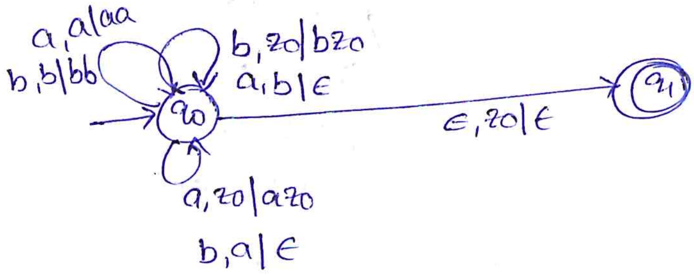
ab ~~b~~ ~~a~~ pop

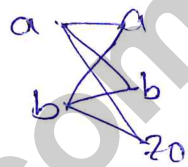
(8)

$\Sigma = \{a,b\}$ $|W_a| = |W_b|$

$w =$ bbbaaa

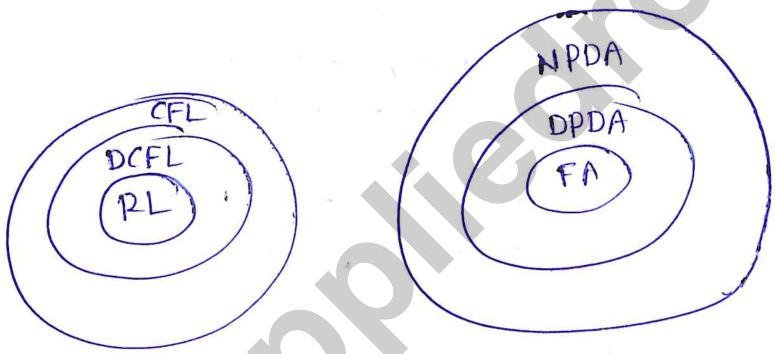
$w =$ baabba





↓↑↓↑↓↑
b a a b b a

RL vs DCFL vs CFL:



DCFL: wxw^R
a^nb^n
$a^mb^n | m<n$

CFL: ww^R
xww^R
ww^Rx

$\boxed{CFG \longleftrightarrow PDA}$

CFG ⟶ PDA:

For every production rule

(1) $A \rightarrow \alpha$ $A \in V$
$\alpha \in (V+T)^*$

$\delta(q, \epsilon, A) = (q, \alpha)$

(2) $\forall a \in T$
$\delta(q,a,a) = (q,\epsilon)$

(1) $S \rightarrow aSB | aB$
$B \rightarrow b$

Let $z_0 = S$
$Q = q$
$q_0 = q$
$F = \phi$

$\delta(q, \epsilon, S) = (q, aSB)$
$= (q, aB)$

$\delta(q, \epsilon, B) = (q, b)$

$\delta(q,a,a) = (q,\epsilon)$ $\delta(q,a,a) = (q,\epsilon)$

$W = aabb \in L(G)$

$\Rightarrow$ is accepted by PDA

$\epsilon aabb, S$

$\cancel{a}abb \quad \cancel{a}SB$

$\epsilon abb \quad SB$

$\cancel{a}bb \quad \cancel{a}BB$

$bb \quad BB$

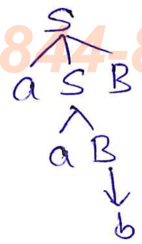
$\epsilon bb \quad BB$

$\cancel{b}b \quad \cancel{b}B$

$\epsilon b \quad B$

$\cancel{b} \quad \cancel{b}$

$\epsilon \quad \epsilon$



(2) $S \to aSA \mid bAB$

$A \to aA \mid a$

$B \to bBA \mid b$

$q_0 = S$

$Q = \{q\}$

$q_0 = q$

$F = \phi$

$\delta(q, \epsilon, S) = (q, aSA)$

$\delta(q, \epsilon, S) = (q, bAB)$

$\delta(q, \epsilon, A) = (q, aA)$

$\delta(q, \epsilon, A) = (q, a)$

$\delta(q, \epsilon, B) = (q, bBA)$

$\delta(q, \epsilon, B) = (q, b)$

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

(3) $S \to AB$

$A \to AS \mid a$

$B \to bBa \mid \epsilon$

$q_0 = S$

$Q = \{q\}$

$q_0 = q$

$F = \phi$

$\delta(q, \epsilon, S) = (q, AB)$

$\delta(q, \epsilon, A) = (q, AS)$

$\delta(q, \epsilon, A) = (q, a)$

$\delta(q, \epsilon, B) = (q, bBa)$

$\delta(q, \epsilon, B) = (q, \epsilon)$

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

(4) $S \to aSB \mid aB$ } GNF $\quad A \to a\alpha \quad a \in T$
$B \to b$ $\quad \alpha \in V^*$
$A \to V$

Phone: +91 844-844-0102

If the grammar is in GNF

$$\boxed{\begin{array}{l} (1)\ \delta(q, a, A) = (q, \alpha) \\ (2)\ \delta(q, a, a) = (q, \epsilon) \end{array}}$$

$z_0 = S$ $\qquad \delta(q, a, S) = (q, SB)$

$Q = \{q\}$ $\qquad \delta(q, a, S) = (q, B)$

$q_0 = q$ $\qquad \delta(q, b, B) = (q, \epsilon)$

$F = \phi$

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

(5) $S \to aAB$
$A \to aBA \mid bAA \mid a$ } GNF
$B \to aBB \mid bAB \mid b$

$z_0 = S;\ Q = \{q\};\ q_0 = q,\ F = \phi$

$\delta(q, a, S) = (q, AB)$

$\delta(q, a, A) = (q, BA)$

$\delta(q, b, A) = (q, AA)$

$\delta(q, a, A) = (q, \epsilon)$

$\delta(q, a, B) = (q, BB)$

$\delta(q, b, B) = (q, AB)$

$\delta(q, b, B) = (q, \epsilon)$

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

Mail: gatecse@appliedroots.com

pumping lemma for CFLS:

For all CFL, L $\exists$ an integer $n \geq 1$

suchthat $\forall z \in L$ and $|z| \geq n$

(i) $z = uvwxy$

(ii) $|vwx| \leq n$

(iii) $|vx| \neq 0$

(or)

$|vx| \geq 1$

then $uv^iwx^iy \in L, \forall i \geq 0$

① . $L = \{a^m b^m c^m \mid m > 0\}$

Let's assume L is CFL

$\exists n \geq 1$

$z \in L$ $|z| \geq n$

$z = a^k b^k c^k$

$3k \geq n$

$a^k \; b^k \; c^k$

$u \; v \; w \; x \; y$

$a^{k-1} \; a \; b^k \; \epsilon \; c^k$

$|vwx| = k+1 \leq n$

hence $uv^iwx^iy \in L \; \forall i \geq 0$

$i=2$ $a^{k-1}a^2 b^k \epsilon c^k \Rightarrow a^{k+1}b^k c^k \notin L$

$\boxed{L \neq CFL}$

② $L = \{a^{m^2} \mid m \geq 0\}$

$\epsilon, a, aaaa, aaaaaaaaa \ldots$

Let L be a Context-free Language. $\exists n \geq 1$

$z \in L \quad |z| \geq n$

$z = a^{k^2}$

pick K such that $k^2 \geq n$

a^{k^2}

u v w x y

a^{k^2-1} a ϵ ϵ ϵ

$|vwx| \leq n$

$1 \leq n$

$|vx| \geq 1$

then $uv^iwx^iy \in L, \forall i \geq 0$

$i = 2 \qquad a^{k^2-1}a^2\epsilon\epsilon\epsilon$

$= a^{k^2+1} \notin L$

$\boxed{L \neq CFL}$

Special case: $\Sigma = \{a\}$ & L is defined over Σ then L is CFL, iff the lengths of strings in L are in Arithmatic progression.

① $L = \{ a^{2n} \mid n \geq 1 \}$ = CFL

aa, aaaa, aaaaaa, ...

2 4 6

② $L = \{ a^{3n+1} \mid n \geq 1 \}$ = CFL

aaaa, aaaaaaa,

4 7 10 ....

③ $L = \{ a^{n^2} \mid n \geq 1 \}$ ⇒ Non CFL

a, aaaa, aaaaaaaaa, ...

1 4 9

④ $L = \{ a^{n!} \mid n \geq 0 \}$ ⇒ Non-CFL

1, 1, 2, 6, 24, 120, ...

⑤ $L = \{ a^p \mid p \text{ is a prime} \}$ ⇒ NOT in AP ⇒ Non-CFL

Closure properties of CFLs:

① **Union:** L_1 : CFL L_2 : CFL

$L_1 \cup L_2$: CFL? → G

$L_1 \rightarrow G_1$, S_1, P_1

$L_2 \rightarrow G_2$, S_2, P_2

$A \rightarrow \alpha$

$A \in V$

$\alpha \in (V+T)^*$

$$\boxed{\begin{array}{l} S \rightarrow S_1 \mid S_2 \\ P_1 \\ P_2 \end{array}} \rightarrow \text{CFG}$$

$S \rightarrow S_1 \rightarrow P_1 \rightarrow L_1$

$S \rightarrow S_2 \rightarrow P_2 \rightarrow L_2$

CFG
✓ Grammar

✓ PDA

RL → FA, RG

② Concatenation:

L_1: CFL ↑ G_1: CFG — S_1, P_1

L_2: CFL ↑ G_2: CFG — S_2, P_2

$w_1 w_2$

G: CFG

$$\boxed{\begin{array}{l} S \to S_1 S_2 \\ P_1 \\ P_2 \end{array}}$$

$S \to S_1 S_2$, $S_1 \xrightarrow{P_1} w_1$, $S_2 \xrightarrow{P_2} w_2$

③ Kleene-Closure:

$w_1 \in L_1$: CFL ↑ G_1: CFG — S_1, P_1

$L_1^* = \{\epsilon, L_1, L_1L_1 \ldots\}$

$G : L_1^*$?

(CFG)

G:

$$\boxed{\begin{array}{l} S \to \epsilon \mid S_1 S \\ P_1 \end{array}}$$ CFG

$S \to S_1 S \to S_1 \to L_1$

$S \to S_1 S_1 S \to S_1 S_1 \to L_1 L_1$

④ Reversal: CFL

$w_1 \in L_1$: CFL ↑ CFG G_1

$w_1^R \to$ CFL? ↑ CFG G_2

$S \to 0S1 \mid 01$

$S \to 1S0 \mid 10$

$\{01, 0011 \ldots\}$

$\{10, 1100 \ldots\}$

5. Homomorphism:

L: CFL

G: $S \to 0S1 \mid 01$

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

$h(0) = ab$

$h(1) = \epsilon$

L': CFL?

$G' \to$ CFG

G': $S \to abS \mid ab$

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

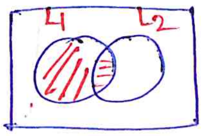
6. Intersection, set-difference: Not closed

$L_1 = \{a^n b^n c^m \mid n \geq 1, m \geq 1\}$ aabbc

$L_2 = \{a^m b^n c^n \mid n \geq 1, m \geq 1\}$ abbcc

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\} \to$ Not CFL (pumping lemma)

Set difference:



$L_1 \cap L_2$

$= L_1 - \underbrace{(L_1 - L_2)}_{L'}$

$= L_1 - L'$

let set difference is closed then is also CFL which is not True.

$\Downarrow$

$L_1 \cap L_2$ is also CFL

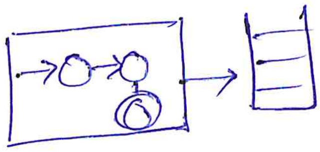
7. Intersection with Regular Languages:

L_1: CFL

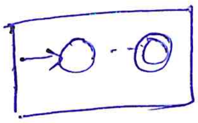
L_2: RL

$L_1 \cap L_2 = CFL?$

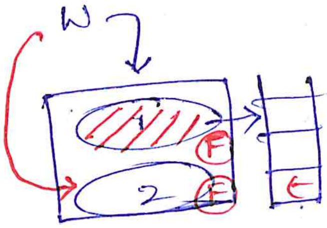
L_1 is CFL



L_2 is RL



FA



w has to be accepted by both FA and PDA then $L_1 \cap L_2$ is CFL

$L_1 = \{a^n b^n \mid n \geq 1\}$ — CFL

$L_2 = \{a^m b^n \mid m, n \geq 1\} \rightarrow$ RL

$L_1 \cap L_2 = \{a^n b^n \mid n \geq 1\} \rightarrow$ CFL

Note: for GATE : Closure properties :

CFLs are closed under :

1. union
2. concatenation
3. Kleene closure
4. Substitution
5. Homomorphism
6. Inverse Homomorphism
7. Intersection with RL
8. Quotient with RL
9. Reverse.

CFLs are not closed under

1. Intersection
2. Complement
3. Set-difference
4. Quotient
5. Inverse Substitution

DCFLs are closed under

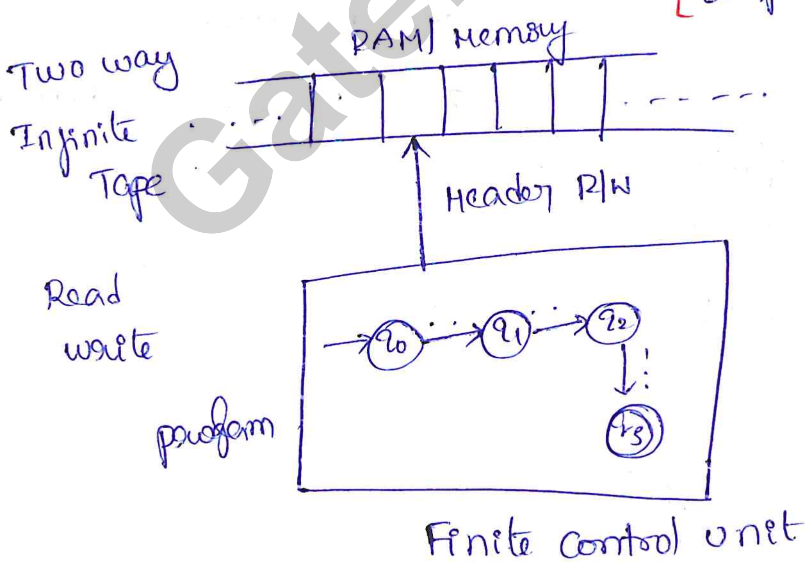
1. Complement → DPDA L_1 L_2
2. Inverse Homomorphism
3. Intersection with RL
4. Quotient with RL

DCFLs are not closed under:

1. Union
2. Concatenation
3. Kleene closure
4. Homomorphism
5. Reversal
6. Intersection
7. Substitution

Turing Machines : An Introduction

{Computer}



Alan Turing

"Father of Computer Science"

Highest Award in Computer Science is Turing Award.

{Most Simple model

what is Computable?

Mathematical Model: 7-tuple

Q: Set of States

Σ: input Alphabet

B: 0 or b Blank Symbol $B \notin \Sigma$ — | B | B | a | a | b | b | B | B | ...

F: Set of final States

q_0: Initial state

Γ : Set of Tape Symbols

$\Sigma \subset \Gamma \quad B \in \Gamma$

$$\delta : \underbrace{Q}_{\text{Current State}} \times \underbrace{\Gamma}_{\text{Tape input}} \longrightarrow \underbrace{Q}_{\text{Next State}} \times \underbrace{\Gamma}_{\text{Replacement Symbol}} \times \underbrace{\{L, R\}}_{\text{Direction}}$$

$x \in \Gamma$

$x \notin \Sigma$

$q_0 \xrightarrow{a,\,X,\,R} q_1$

Eg1: $\{a^n b^n c^n \mid n \geq 1\}$: pumping lemma of CFL $\rightarrow$ No PDA

$\exists$ TM

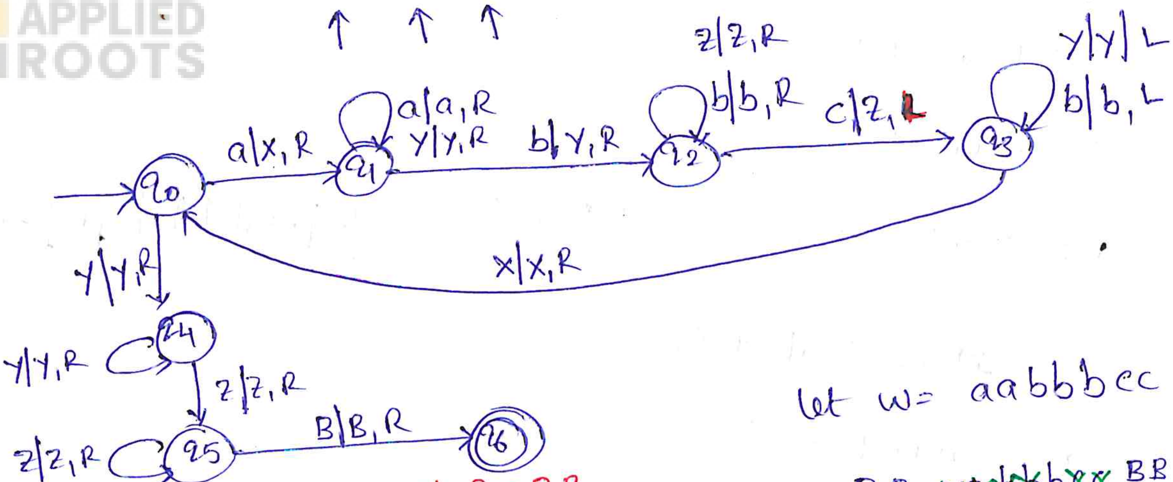
aabbcc $\rightarrow \boxed{\text{C-program}}$

TM $\approx$ Computer

BB $\overset{X}{\cancel{a}}\overset{X}{\cancel{a}}\overset{X}{\cancel{a}}\,\overset{Y}{\cancel{b}}\overset{Y}{\cancel{b}}\overset{Y}{\cancel{b}}\,\overset{Z}{\cancel{c}}\overset{Z}{\cancel{c}}\overset{Z}{\cancel{c}}$ BB

BB $\overset{X}{\cancel{a}}\overset{X}{\cancel{a}}\,\overset{Y}{\cancel{b}}\overset{Y}{\cancel{b}}b\,\overset{Z}{\cancel{c}}\overset{Z}{\cancel{c}}$ BB $\rightarrow$ Invalid String

X Y Z
BB aa bb cc BB



BB X a Y b Z c BB

Let w = aabbbcc

BB ~~aabbb~~b~~cc~~ BB
X X Y Y b Z Z

$q_4 \xrightarrow{b}$ No transition

Reject

BB X X Y b Z c BB
Y Z

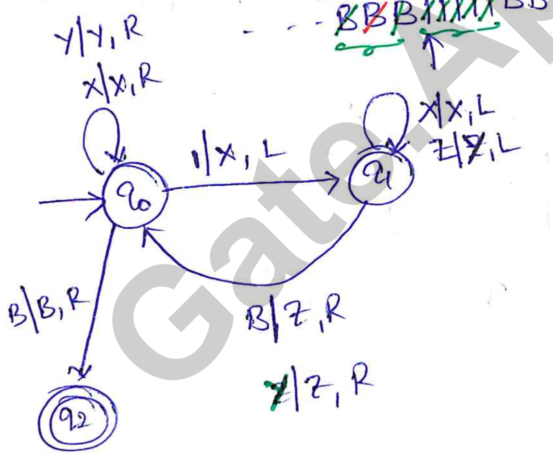
BB X X Y Y Z Z BB

BB X X Y Y Z Z BB

11111 ⟶ 101 Base$_2$

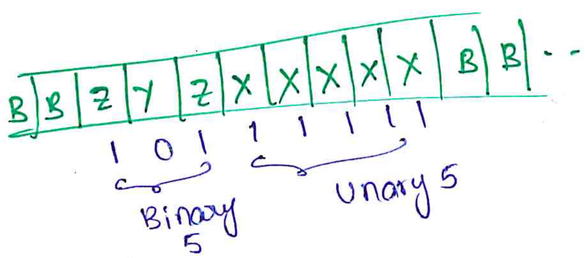
Eg 2: Convert unary to Binary

BBB11111BBB......
Z Z Y X X X X X



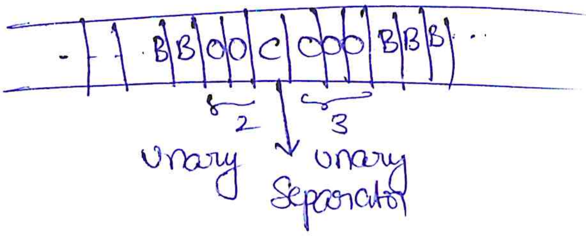
U	B	
1	1	: Z
11	10	: ZY
111	11	: ZZ
1111	100	: ZYY
11111	101	: ZYZ

{Z = 1, Y = 0}

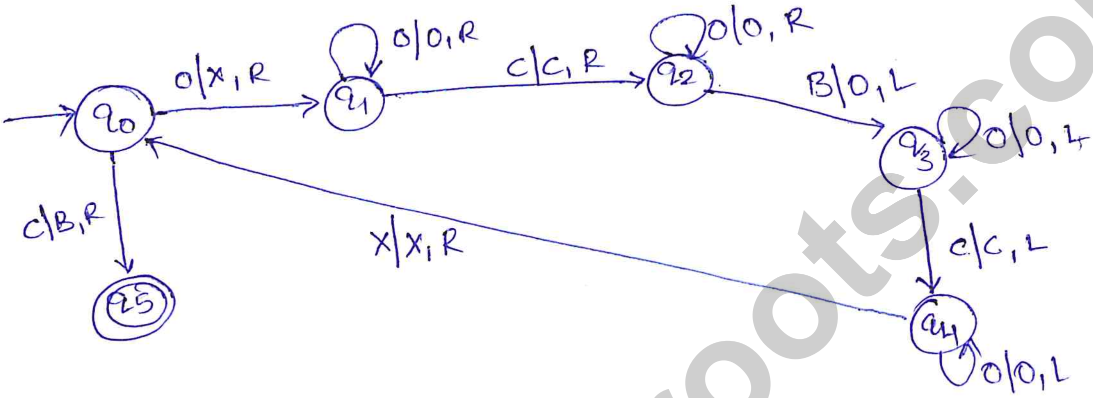


Eg 3: TM for unary addition

input:



output: ..BB 0 0 0 0 0 BB.... unary 5



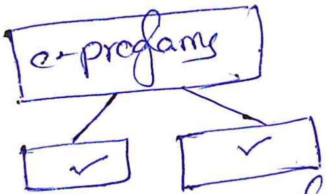
..BB ~~00~~ C 000 ~~BB~~B...

Input and o/p machine.

..BB XX C $\boxed{000\ 00}$ BB...

C → B unary 5

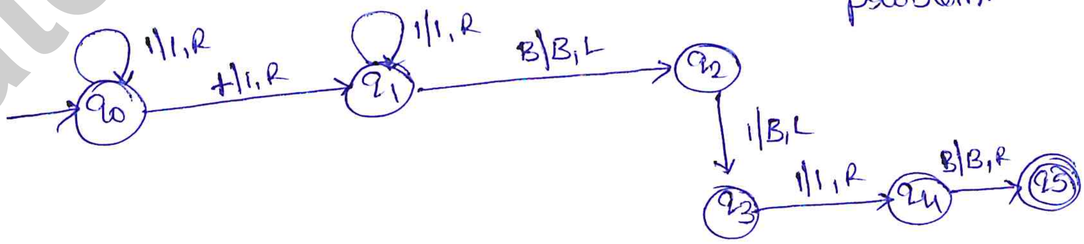
TM for unary addition



disjoint programs solve the same problem.

input: ....BB 11+111BBB....

....BB 111 11BBB...



Properties and Types of TM :

TM as a transducer : math-funs : addition, subtraction, Multiplication

inputs → f → output

factorial

e^x

⋮

recursive funs: fib(n)

= fib(n-1) + fib(n-2)

→ Languages $\{a^n b^n c^n \mid n \geq 1\}$

Simulator (Univ TM)

program

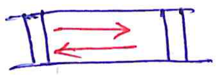
Halting & Acceptance : Stop = Halt

→ $\boxed{\text{Undecidable}}$

① Final state + Halt → Accept

$q_0 \xrightarrow{a|x,R}$

② Non-final + Halt → Not-accept



③ ∞-loop (or) No Halt → cannot conclude

while(1)
{
C:
}

undecidable $\left\{ \begin{array}{l} \boxed{w \in L(TM)} \\ \boxed{w \notin L(TM)} \end{array} \right.$ (or)

DTM & NTM : DTM = NTM (same power)

↑ Default

↑ More steps | transitions

DFA = NFA

DPDA ≠ NPDA

↓ DCFL CFL

DTM: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

$q_0 \xrightarrow{a|x,R} q'$

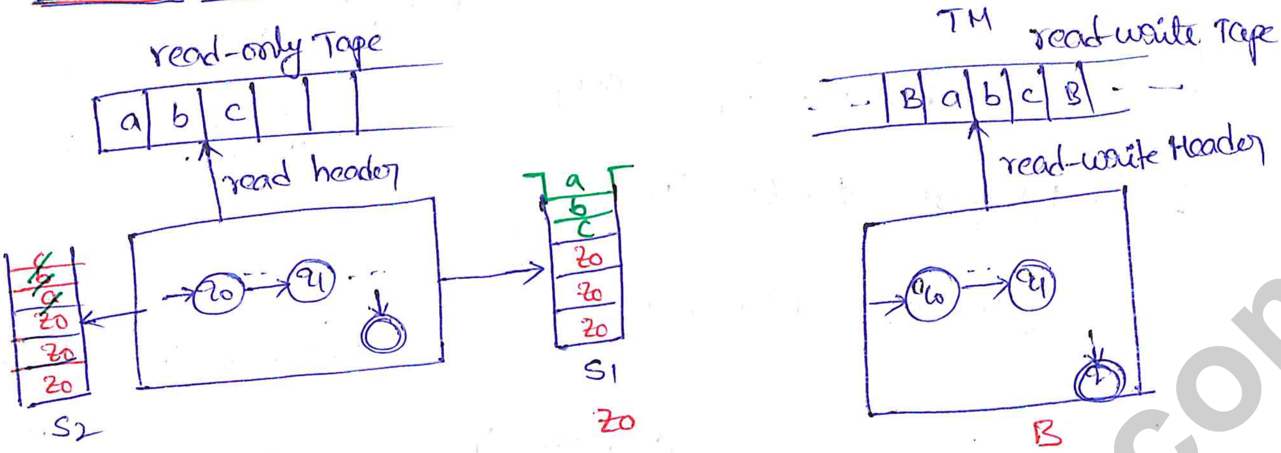
NTM: $\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$

$q_1 \xrightarrow{a|x,R} q'$

$q_1 \xrightarrow{a|y,L} q_1$

$(q,a) \longrightarrow \{(q', x, R), (q_1, y, L)\}$

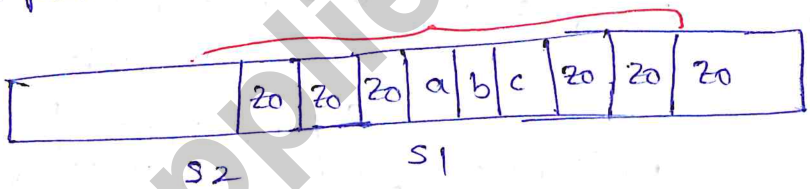
PDA with 2-stacks:



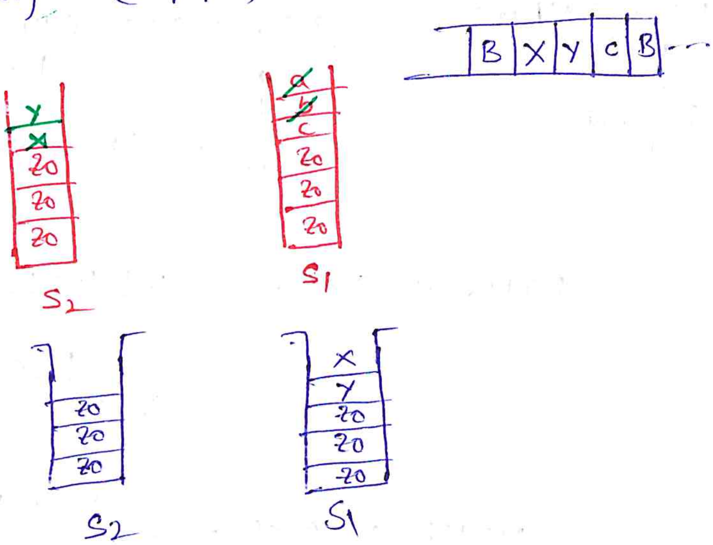
2- Stacks

PDA with 2-stacks = TM ?

1. PUSH z_0's onto both of the stacks
2. push input word onto S2
3. pop and push w onto S1

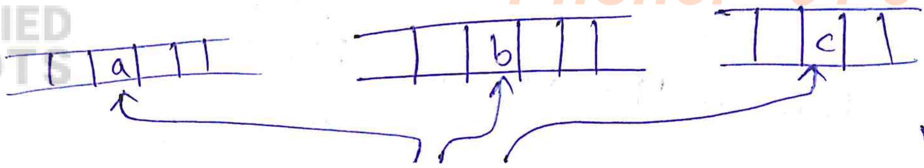


TM $\delta(q,a) = (a|x,R)$ $\delta(q,b) = (b|y,L)$



$\delta : Q \times \Gamma_R \times \Gamma_L \longrightarrow Q \times \Gamma_R \times \Gamma_L$

Multi-tape TM:

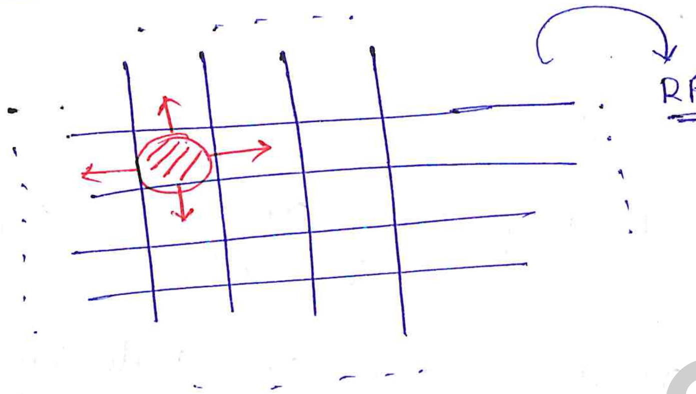


= DTM

More Steps | Transitions

$\delta(q, a, b, c) = (q', x, y, z, L, R, L)$

Multi-dimensional TM:

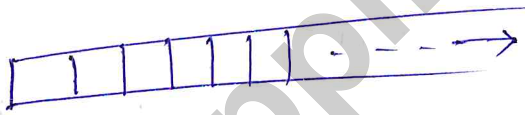


RAM

= TM

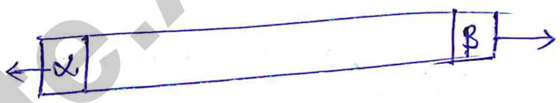
$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, U, D\}$

Semi-Infinite TM:



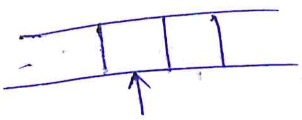
∞-Tape = TM

LBA:



LBA < TM

TM with stay-option: = TM



$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$

Stay

TM recognizable Languages:

FA ⟷ RL

PDA ⟷ CFL

TM ⟷ ?

↳ Recursive Language | Rec Sets (REC)

↳ Recursively Enumerable Languages. (REL)

↳ Chomsky [Unrestricted Grammar]

$\alpha \to \beta$

Recursive Language: (REC)

L   TM

$\forall w \in L$, TM accepts $w \Rightarrow$ TM reaches a final state & halts.

$\forall w \notin L$, TM reaches a non-final state & halts

⇓

TM rejects w

Recursively Enumerable Language (REL):

L   TM

$w \in L$ then TM reaches a final state & halts

⇓

w accepted

$w \notin L$ then TM < reaches a non-final state & halts (or) ∞-loop

$\boxed{\text{does } w \in L?}$ membership property

If L is a Recursive Language, then we can decide on the membership property.

If L is REL; $w \to$ TM < Final + halt / Non-final + halt / ∞-loop

then we cannot definitely decide if $w \in L$ or not?

Membership is undecidable for REL

Decidable for Recursive Languages.

Closure properties of REC :

TM → final + halt = A

Non-final + halt = R

∞-loop = ∞

① Union of REC

$L_1 = L(M_1)$, $L_2 = L(M_2)$

($L(M_1)$: A, R; $L(M_2)$: A, R)

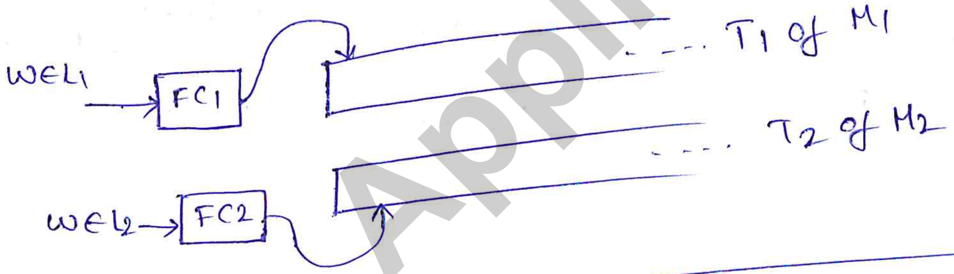
Is $L_1 \cup L_2$ = ?

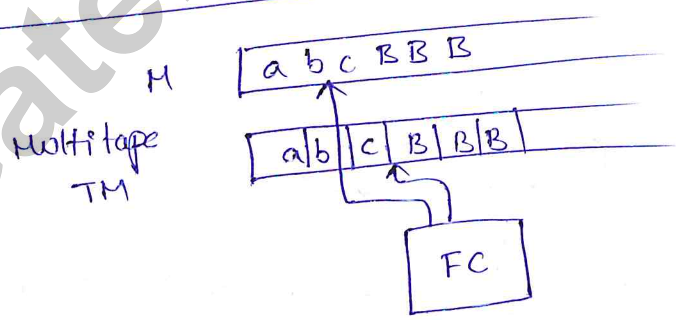
L(M)

A R

STM ≈ semi-∞ TM

≈ Multitape TM





① Copy w from T_1 to T_2

② Parallely Simulate M_1 on T_1, Simulate M_2 on T_2

③ M_1 < A, R

M_2 < A, R

Either M_1 or M_2 Accept ⇒ then w is accepted.

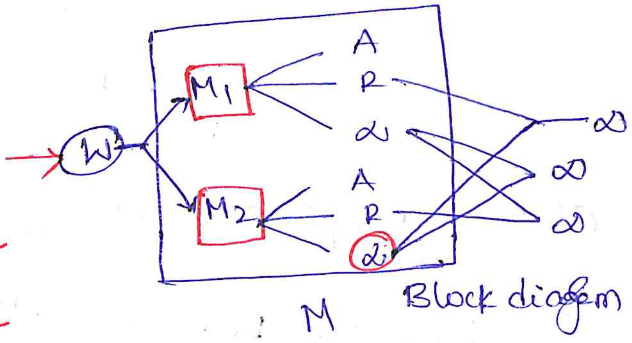
If both M_1 & M_2 Reject w then w is Rejected.

① ⓑ **Union of REL :** Closed

REL $\begin{cases} L_1 = L(M_1) \\ L_2 = L(M_2) \end{cases}$

$w \in L_1 \cup L_2 \longrightarrow A$ ✓

$w \notin L_1 \cup L_2 \longrightarrow R$ ✓, ∞ ✓



any one accepts – Accept

Both Rejects – Reject

② **Intersection:**

$L_1 \quad L_2$

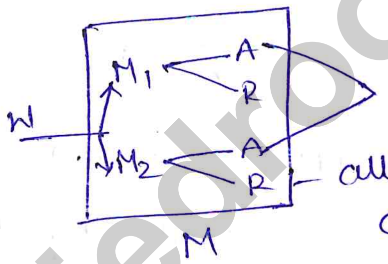
$\downarrow \quad \downarrow$

$M_1 \quad M_2$

REC

$\Downarrow$

closed under intersection

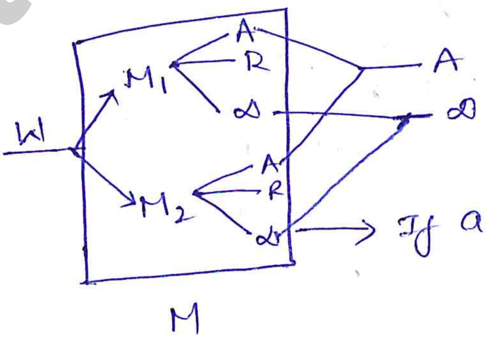


$W - M_1 - R \Rightarrow W \notin L_1$

$W - M_2 - A \Rightarrow W \in L_2$

} Rejected

Intersection of REL :



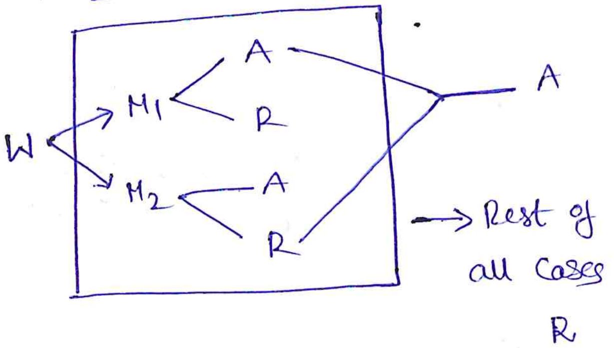
If any one Rejects – R

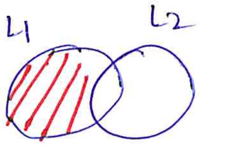
RELs are closed under intersection.

③ Set difference:

$L_1 - L_2$

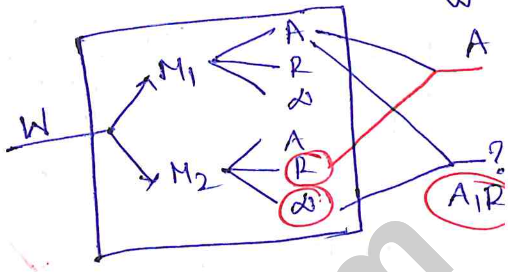
REC:





$L_1 - L_2$

REL: Not closed under set-difference



$w \in L_1 - L_2$

$\Rightarrow w \in L_1$ & $w \notin L_2$

If M_1 accepts w, and M_2 stucks for w loops

For few/some words we cannot decide it is accepted (or) Rejected

$L^c = \Sigma^* - L$

REL REL

Not closed

$\boxed{\text{undecidable for RELs}}$

Complement :-

$L^c = \Sigma^* - L \rightarrow$ Recursive

↳ Reg Lang

⇓

Recursive

$L^c = REC - REC$

∴ Recursive Languages are also closed under Complement operation.

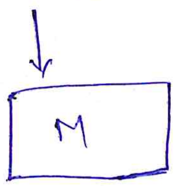
④ Concatenation: (REL)

$x \in L_1 \ (M_1)$

$y \in L_2 \ (M_2)$

let $w = xy$

$w = 1011110$



$x = 1011$

$y = 1110$

break-point

① **NTM :** "guess" and create non-deterministic transitions for each possible break point.

$\boxed{M}$

can we construct a TM or not?

W = 10 11 1110

M_1 M_2

A R ∞ A R ∞

A & A ⟶ A

R & R ⟶ R one of them Reject

∞ & ∞ ⟶ ∞

W = 1011 1110 break point

x = 1011

y = 1110

④ ⓑ **Concatenation of REC :** concatenation is closed for Recursive Languages.

ω = x y {Try all possible break points

101 11

M_1 M_2

A R A R

M_1 M_2

A & A ⇒ A

R & – ⇒ R

– & R ⇒ R

⑤ **Kleene Closure:** ⟶ REC, REL are closed

$L_1^* = \{\epsilon, W_1, W_1 W_2, \omega_1 \omega_2 \omega_3, \ldots\}$

try multiple break points & all possible combination of positions

$L_1 = \{101, 11, 10\}$

W = 101 11 10

1011110

$w \in L_1^*$

101 11 10 ⟶ M_1 – A

M_1 M_1

A A

if all strings are accepted ⇒ accept

Any string is rej ⇒ Reject

All strings ∞ ~~then~~ then ∞ loop

Recursive Languages:

Closed under:

1. Union
2. Concatenation
3. Intersection
4. Complement
5. Inverse-Homomorphism
6. Reverse
7. Intersection with RL
8. Kleene closure
9. Set-difference

Not-closed under

1. Homomorphism
2. Substitution
3. Quotient with RL

Recursively Enumerable Languages:

Closed under:

1. Union
2. Concatenation
3. Intersection
4. Substitution
5. Homomorphism
6. Inverse-Homomorphism
7. Intersection with RL
8. Quotient with RL

Not closed under

1. Set-difference
2. Complement

Note: If L and L^c are Recursively Enumerable languages then L must be Recursive.

(Q) $L, L^c = \Sigma^* - L$

valid { (a) L & L^c are REC

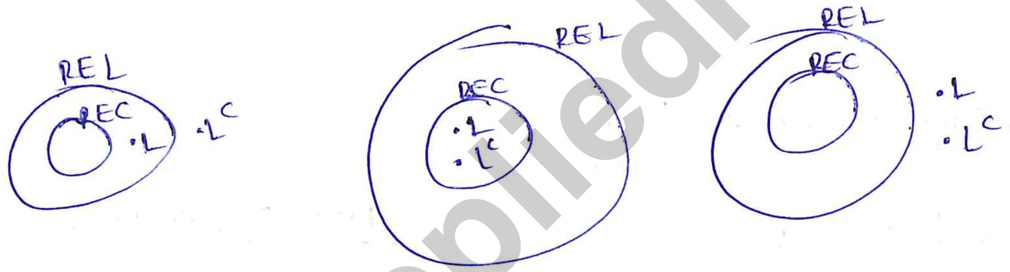
(b) L and L^c are not-Recursively Enumerable

(c) L is REL & L^c is not REL }

✗ (d) L is REL & L^c is also REL

⇓

NOT True for all Languages.



Linear Bounded Automata & Context Sensitive Languages:

LBA ⟷ CSL ⟷ CSG

Chomsky Hierarchy of Languages.

LBA = NTM + Bounded Tape + Linear bounded

⇓

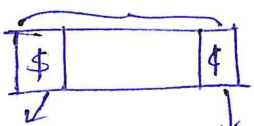
(w)

$\alpha \longrightarrow \beta$

$|\alpha| \leq |\beta|$

$\alpha, \beta \in (V+T)^+$

Read-write Tape:



Left end Marker

Right end marker

linear in $|w|$

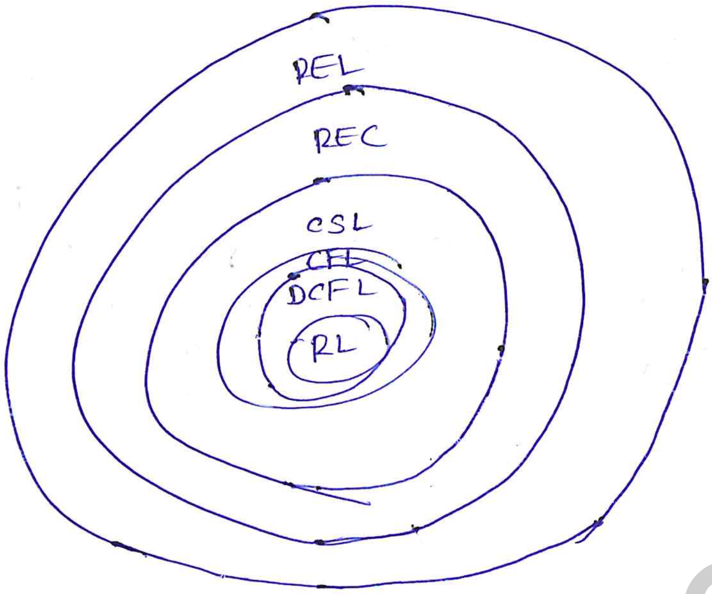
$a|w| + b$

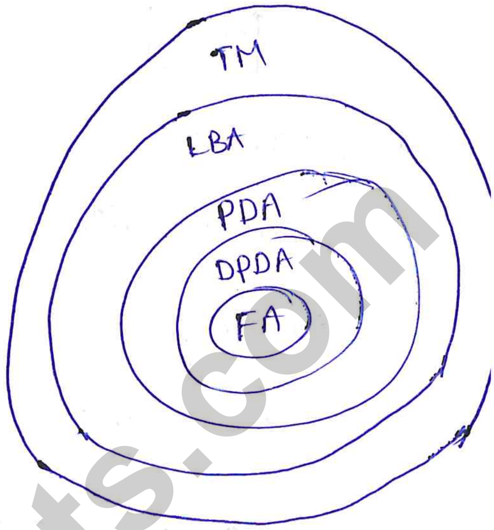
$a|w|^2 + b|w| + c$

$(Q, \Sigma, \Gamma, B, \$, ¢, q_0, F)$

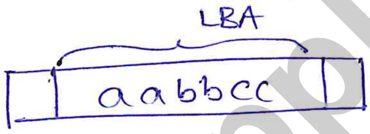
$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$

CSG & CSL





$L = \{\underbrace{a^n b^n c^n}_{3n} \mid n \geq 1\} \longrightarrow CSL$



Closure properties of CSL:

CSLs are closed under

1. positive closure (L^+)
2. union
3. Intersection
4. concatenation
5. Reversal
6. Inverse Homomorphism
7. Intersection with RL

Not closed under

1. Homomorphism
2. Substitution

Undecidability & Computational classes :- Real-world implications

NP-Complete
NP-Hard
P

↳ Halting problem

$\boxed{T.M}$

(i) L That is non-Recursively Enumerable

(ii) L is REL but not REC

Deep & Tricky

Reducibility

Encoding a TM: (Using binary Strings)

music, Audio, Video, Image → Computer ⇓ Binary

$Q, \Sigma, \Gamma, q_0, F, \{L, R\}, \delta, B$

No q_0:
- q_1: Initial State
- q_2: only Final state
- $q_3, q_4 \ldots$: other States

No x_0:
- $x_3 : B$
- $x_1 x_2 x_4, x_5 \ldots\ldots$ other Symbols

no $D_0 \longrightarrow$ L: D_1 R: D_2

$$\delta(q_i, x_j) = (q_k, x_l, D_m)$$

$$(q_i, x_j, q_k, x_l, D_m)$$

$i, j, k, l, m > 0$

⇓

$W = \underbrace{0^i 1 0^j 1 0^k 1 0^l 1 0^m}_{1^{st}\ \text{transition}}$ 11 $\underbrace{\qquad}_{2^{nd}\ \text{transition}}$

$\in (0+1)^*$

del

TM : binary string = binary number$_{base 2}$ = decimal number

0^{th} TM, 1^{st} TM, 2^{nd} TM, ------. TM_i

Not every binary String is a TM   11001 = not a valid TM

$(2)_{10} = (10)_2 = 10 \neq$ any valid TM binary encoding

↳ TM [→ (q)] $L(TM) = \phi$

$TM_i = \{TM_0, TM_1, \ldots\}$

Diagnolization Language : A Non-REL   Ld ⇒ No TM to recognize $w \in Ld$

$Ld = \{ w_i \in \{0+1\}^* \mid w_i \notin L(M_i) \}$ ← Super Creative ← pougf : Genius

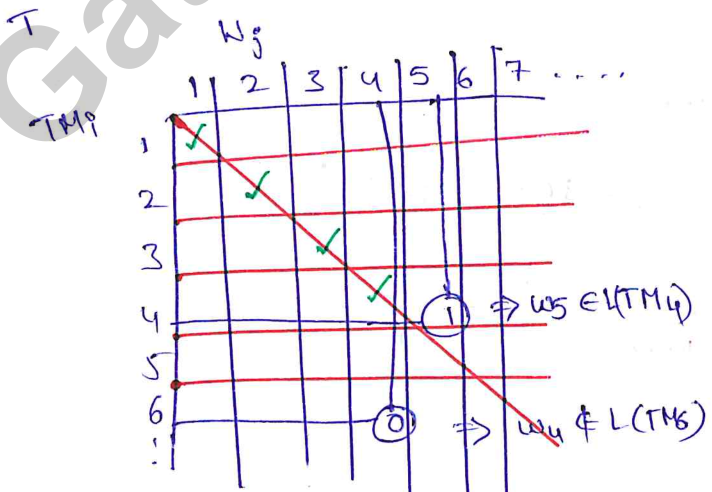
i : Integer → $base_2$ → binary String → M_i
→ $Word_i (w_i)$

Set of Strings which are not accepted by a machine represented by the same String.

Is Ld a REL?
⇓
if $w \in Ld$ then ∃ TM M such that $w \in L(M)$

$w \in Ld$ → Halt + Accept

$w \notin Ld$ → Halt + Reject or ∞



diagnolization

3 ⇔ 11 < w_3, TM_3

$d = diag(T)$

= | 1 | 0 | 1 | ... | 1 |
|---|---|---|---|---|

1 if $w_i \in L(TM_i)$
0 if $w_i \notin L(TM_i)$

Binary Matrix

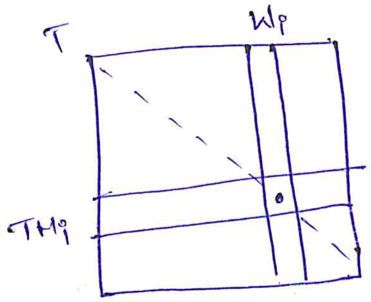
d' = | 0 | 1 | 0 | . | . |
|---|---|---|---|---|

$d = diag(T)$

$d' = $ [| | | | | i]

↳ 1 if $w_i \notin L(TM_i)$

↳ 0 if $w_i \in L(TM_i)$



$d'[i] = 1 \Leftrightarrow w_i \notin L(TM_i)$

$$\boxed{L_d = \{ w_i \mid d'[i] = 1 \}}$$

(*) Is L_d is REL? $\boxed{\text{Non-REL}}$

Every TM corresponding to i TM_i

⇑

$\Leftrightarrow$ Is there a TM that accepts L_d?

⇑

$\Leftrightarrow$ Is there a Row in T, which is equal to d'-vector.

⇑

Let ith row be the row in T, which is equal to d'-vector

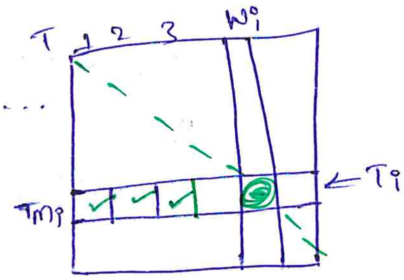
does not exist

Proof by Contradiction

$\Rightarrow T_i = d'$

$\Rightarrow T_i[j] = d'[j] \quad \forall j = 1, 2, 3 \ldots$

$\Rightarrow T_i[j] = (d[j])'$



$\Rightarrow$ for $j = i$

$T_j[j] = d[j] = (d[j])'$

	1	0
contradiction	0	1

Universal Turing Machine: (L_u) Universal Language

TM: program   **UTM:** General purpose Computer

UTM: TM that takes a TM-Code/binary representation of a TM (M) and w as inputs and it accept the inputs (word) iff $w \in L(M)$

$L = \{a^n b^n c^n \mid n \geq 1\}$
aabbcc

Input to a UTM: Code for M 111 W $\in (0+1)^*$

$w \in (0+1)^*$

Binary String

delimiter/Separator

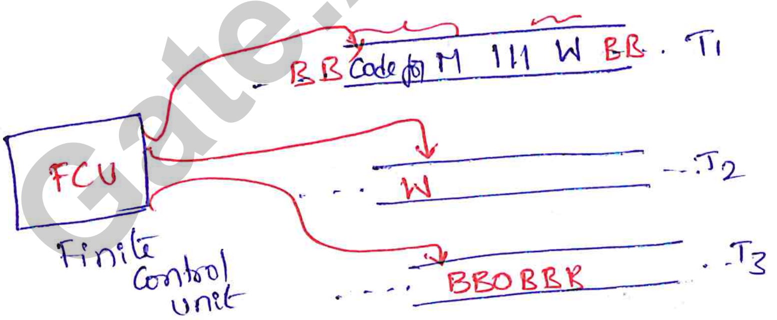
on some inputs UTM
↳ accepts + Halt
↳ Reject (NF + H)
↳ ∞-loop

$L_u = \{ L(U) \}$

⇓
Universal Language.

U: UTM

Working of UTM U:



to Simulate the Tape of M

Current State of M during Simulation

① Check if M is a valid TM
→ if NO $L(M) = \phi$
⇒ $W \notin L(M)$
UTM halts + Reject

→ Yes : Continue

② Copy W from T_1 to T_2

③ Initialize T_3 with initial state of M

④ Simulate TM M using T_1, T_2 & T_3

$$\delta(q_1, 0)$$
$$\Downarrow$$
$$\delta(q_1, x_1) = (q_2, x_2, R)$$

$x_1 : 0$

$x_2 : 1$

$x_3 : B$

→ Scan the code for M

11 0101 0010010011

$$\delta(q_2, 0)$$
$$\Downarrow$$
$$\delta(q_2, x_1) = ?$$

11 00101

(F+M)

⑤ If M accepts W

then UTM also accepts input

$\boxed{\text{Code for M 111 W}}$

① Is L_u REC (or) REL (or) Non-REL?

$L_u = L(UTM)$ ∃ a TM = UTM that accepts L_u

UTM = TM

$\exists$ TM that halts & accepts L_u

(Total TM)

decider

Intutive: M 111 W $\in L_u$?

$\Downarrow$

Simulating W & M on UTM

$\Downarrow$

∞

If M is a TM that does not halt on w

Reduction: [Most useful ideas], Algorithms (P, NP...)

Let's assume $\boxed{L_u \text{ is REC}}$

$\Rightarrow$ Decide if $w \in L_u$ using a TTM (M_u) (or) Algorithm (A_u)

let's take L_d (is Not REL) $\longrightarrow$ diag

Instance of Membership in L_d

problem: $w \in L_d$?

(1) Is $w \notin L(M_w)$?

Algorithms: (1) If M_w is not-valid

$\Rightarrow L(M_w) = \phi$

$\Rightarrow w \notin L(M_w)$

$\Rightarrow w \in L_d$

(2) If M_w valid

W 111 W Construct

TM$_w$ input

Reduction Alg

Red-TM

$w \notin L_d$

W 111 W $\longrightarrow$ M_U (TTM)

A $\longrightarrow$ $w \in L(M_w)$

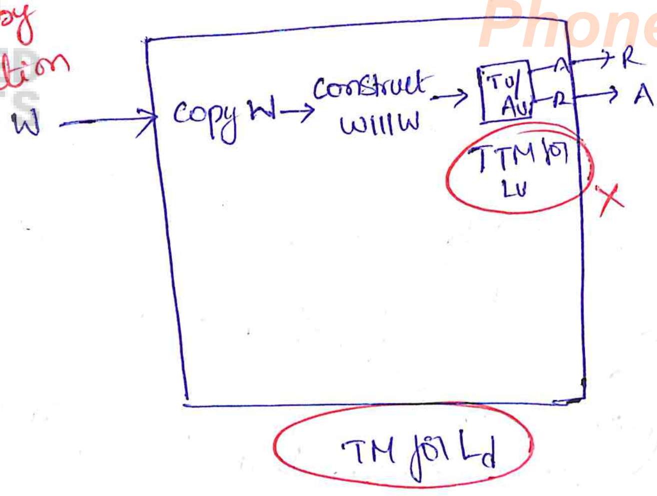
R $\longrightarrow$ $w \notin L(M_w)$

$\downarrow$

$w \in L_d$

$A \rightarrow B$ Reduction-diagram

proof by contradiction

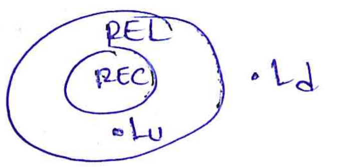


- If $\exists$ TTM for Lu then I can construct a TM which accepts Ld.

{we know that $\nexists$ TM that accepts Ld}

<u>Reduction</u>:

Instance of $w \in Ld$
↓ Algorithm / TTM Reduction
Instance of $w \in Lu$
↓
assumption $\exists$ Solution
⇕
$\exists$ TTM



REC - decidable

If there is a reduction from P_1 to P_2

instance of P_1
↓ Alg
P_2

Then

(1) P_1 is undecidable $\Rightarrow$ P_2 is also undecidable.

(2) P_1 is Non-REL $\Rightarrow$ P_2 is also Non REL

$w \in Ld$?

<u>Halting problem</u>: Does a TM, M halt on input w?

$\boxed{\text{Undecidable}}$

⇕

Is $w \in L(M)$

TM → A F+halt ; R NF+Halt ; ∞

$w \in L(M) \rightarrow$ halt + Accept

$w \notin L(M) \rightarrow$ R or ∞

Is $\langle M \, 111 \, W \rangle \in L_U$ — L_U is REL & not REC

$\Updownarrow$

[Undecidable]

Problems ~ Languages

P	L
① Does a TM accepts it's own code? $\longleftrightarrow$	Membership in L_d $\quad w \notin L(M_w)$
② Does a TM halt on W (Y / N) $\longleftrightarrow$	Membership in L_U
③ Does a given G contain a cycle (Y / N) $\longleftrightarrow$	$L_C = \{ w \mid G_w \text{ contains a cycle} \}$
G :→ Encode graph using bin-strings (w)	$w \in L_C$? TM? $\quad w \in \{0,1\}^*$

∃ CU $\overset{TM}{(\delta)}$ ⟶ Graph

$L_d \rightarrow$ Non-REL $\quad L_U \rightarrow$ REL

L_e and L_{ne} :

$L_e = \{ M \mid L(M) = \phi \}$ — M: binary-rep of TM

$M \in (0+1)^*$

$L_{ne} = \{ M \mid L(M) \neq \phi \}$ — M: binary string; L(M): TMM

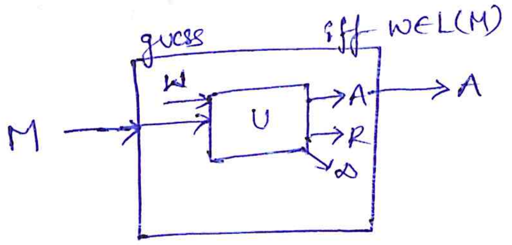
L_{ne} — Non-REL? / REL? / REC?

$$\left\{ \begin{array}{l} L_e = L_{ne}^c = \overline{L_{ne}} \\ L_{ne} = L_e^c = \overline{L_e} \end{array} \right\}$$

↳ Closure properties of REL & REC

Is L_{ne} a REL? $\Rightarrow$ does $\exists$ TM s.t $L_{ne} = L(TM)$

NTM ~ DTM



iff $L(M) \neq \phi$

NTM-M_{ne}

$\Rightarrow$ $\boxed{L(M_{ne}) = L_{ne}}$

$w_1 w_2$

{ similar strategy to prove closure prop of concatenation }

Is L_{ne} a REC-Language? $\Leftrightarrow$ $\exists$ a TTM, TM s.t

$L(TM) = L_{ne}$

Guess: No

Prove: Reduction

$\nexists$ a TTM

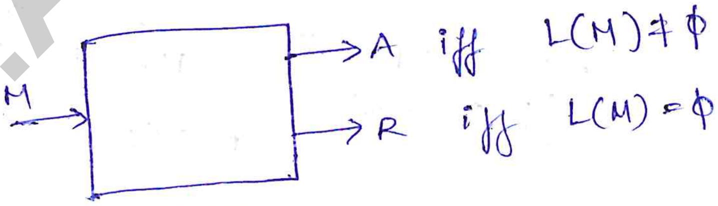
L_U: not REC but REL: undecidable language.

$\left\{ \begin{array}{l} L_U \xrightarrow{\text{Red. Alg}} L_{ne} : \text{let } \exists \text{ TTM} \\ M111w \end{array} \right.$

let's assume L_{ne} is REC language.

$\Leftrightarrow$ $\exists$ an algo/TTM for L_{ne} to determine if $L(M) \neq \phi$

$\Leftrightarrow$

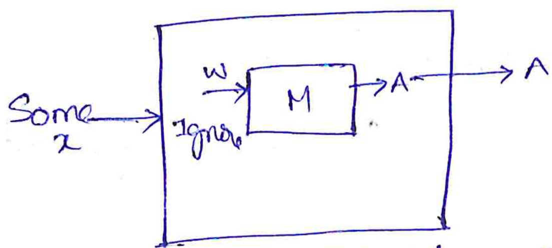


} Hypothetical Alg

TTM_{ne}

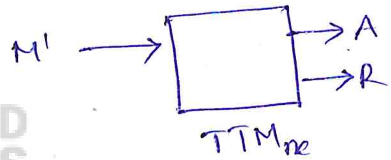
Reduction Algorithm: $M111w \in L_U \longrightarrow$ word in L_{ne}

1



M' accepts some x iff $w \in L(M)$

TM: M' $\longrightarrow$ Represent M' as a bin-string.

② 

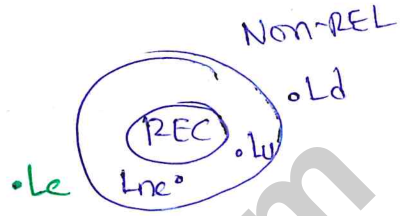
iff $L(M') \neq \phi$

iff $L(M) = \phi \rightarrow w \notin L(M)$

Assumption is incorrect.

$\nexists$ a TTM$_{ne}$ $\Rightarrow$ L_{ne} is not REC

$\exists$ a TM$_{ne}$ $\Rightarrow$ L_{ne} is REL



$L_e = \overline{L_{ne}} = L_{ne}^c$

if L_e is REL

$\overline{L_e} = L_{ne}$ is REL } L_e & L_{ne} are REC

RELs are not closed under Complement
REC are closed under Complement

$\boxed{L_{ne} \text{ is not REC}}$

$\Rightarrow$ $\boxed{L_e \text{ is non REL}}$

L_e & L_{ne} : Special case of Rice Theorem

Note: **Reduction :**

① If $\exists$ a Reduction from P_1 to P_2, then (P_1: Lu, P_2: Lne)

ⓐ if P_1 is undecidable, then P_2 is also undecidable. (P_1: Lu, P_2: Lne)

ⓑ if P_1 is non-REL then P_2 is also a non-REL ($P_1 \downarrow$ Ld, $P_2 \downarrow$ Le)

Note for GATE:

- Covered the most important ways to prove in earlier videos.

- Previous Gate Questions: Mostly about properties & applying them.

- About the proofs, please refer text books → Hopcroft, Ullman; Micheal Sipser

Property of a REL:

= Set of REL that satisfy a property of P

P_{CFL} = Set of REL that are also CFL $\quad P_{CFL} = \{L_1, L_2 \ldots\}$ → REL & P

P_{INF} = Set of REL that are ∞-Languages

Trivial property:

$P = \phi$ or P = all Set of recursively enumerable Languages.

Rice Theorem: Every non-trivial P of REL is "undecidable"

1960's

undec $\Leftrightarrow$ Not REC

$\Leftrightarrow$ REL & non REL

L_e, L_{ne} → REL, → Non REL

$L_U \xrightarrow{\text{Reduce}} L_P$?

(undec)

$P = \{L_1, L_2 \ldots L_i \ldots\}$

$L_1 \downarrow M_1$, $L_2 \downarrow M_2$, $L_i \downarrow$ REL & P satisfy $\downarrow$ M_i

$L_P = \{M_i \mid L_i \in P\}$

Applications:

① P: whether a Language accepted by a TM is Empty

$L_e = \{ M \mid L(M) = \phi \}$ undecidable

↳ Non-Rel

② P: Lang accepted by TM is Finite

$L_p = \{ M \mid L(M) \text{ is Finite} \} \longrightarrow$ Undecidable

③ P: Language accepted by a TM is Regular → undecidable

P: Language accepted by a TM is CFL → Undecidable.

④ whether/Does a TM have 5 States? "Decidable"

Transition Graph/Table/Encoding

we can write an algorithm to count the number of states in a TM.

⑤ Does ∃ an input w s.t TM makes atleast 5 moves? < Y / N

→ cannot apply RICE Theorem.

Algo ⇓ Dec

Non-dec Reductions

$L_u \rightarrow \ldots$

$L_d \rightarrow \ldots$

W = ...01011 ... ⇓ 2^5

...0111 ... ⇒ 2^4

⋮

... ε ... ⇒ 2^0

$2^6 - 1$

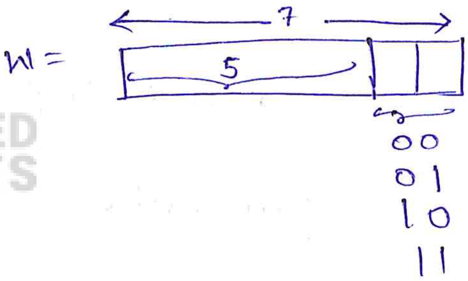
→ TM → Halt → Halts in all case

→ Non-Halt

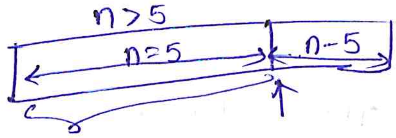
>5

W = |← 5 →| 0 | 1 |

|← 6 →|



Decidable problem



Post's correspondence problem: (PCP) : Strings $\Sigma = \{0,1\}$

→ Undecidable ⟶ TM

L_u

$|A| = |B|$

Corresponding pair: (w_i, x_i)

List

	A	B
i	w_i	x_i
1	1 w_1	111 x_1
2	10111 w_2	10 x_2
3	10 w_3	0 x_3

w_2, w_1, w_1, w_3 x_2, x_1, x_1, x_3

101111110 101111110

Equal no of Strings { A: List of strings of Σ
B: List of strings of Σ

$w_2 w_1 w_1 w_3 = x_2, x_1, x_1, x_3$

101111110 10 111 1110

2, 1, 1, 3

A: List of words in $\Sigma^* = w_1, w_2 \ldots w_k$

B: " " $= x_1, x_2 \ldots x_k$

PCP has a solution iff $\exists$ a sequence of indices

$i_1, i_2, \ldots i_m$

set

$w_{i_1} w_{i_2} w_{i_3} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}$

Eg 2:

	A	B
i	w_i	x_i
1	10	101
2	011	11
3	101	011

No Solution

① $i_1 = 1$ 10 101

② $i_2 = 1$ 1010 $\times$ 101101

$i_2 = 2$ 10011 $\times$ 10111

$i_2 = 3$ $\boxed{10101}$ $\boxed{101011}$ (1, 3, 3, 3 ...)

③ $i_3 = 1$ 1010110 101011101 $i_3 = 3$

$i_3 \neq 2$

PCP: PCP is REL $\rightarrow$ L_{ne} : NTM guessing W

$\rightarrow$ NTM guessing the seq of $i_1, i_2, \ldots$

PCP is not REC $\rightarrow$ $\boxed{\text{pcp is Non-decidable}}$

$L_U \xrightarrow{\text{Reduce}} \text{PCP}$

$L_U \xrightarrow{\text{Reduce}} \text{Modified pcp} \xrightarrow{\text{Reduce}} \text{PCP}$

$L_U \downarrow$ $\boxed{\text{Non-dec}}$ PCP $\downarrow$ $\boxed{\text{Non-dec}}$

Logically:

$\nexists$ a seq $i_1, i_2 \ldots i_m$

$\Downarrow$

∞-loop

Decision

Problems about Languages:

Phone: +91 844-844-0102

① Membership problem ? $<^{Y}_{N}$ does $w \in L$? $<^{Y}_{N}$

② Emptiness : Is $L = \phi$ $<^{Y}_{N}$

③ Finiteness : Is $|L|$ finite? $<^{Y}_{N}$

④ ~~Equal~~

Equivalence : Is $L_1 = L_2$?

Is $L(M_1) = L(M_2)$?

Is $L(G_1) = L(G_2)$?

Is $L(r_1) = L(r_2)$?

⑤ Intersection Empty $L \sim G \sim M$-regexp

Is $L_1 \cap L_2 = \phi$

⑥ Totality | Completeness Is $L = \Sigma^*$?

⑦ subset : Is $L_1 \subseteq L_2$?

⑧ Intersection - Finiteness : Is $|L_1 \cap L_2|$ Finite?

⑨ Co-finiteness : given L, Is $|\bar{L}|$ finite?

⑩ Regularity : Is L is a Regular lang?

⑪ Ambiguity problem: $L \sim G$

Is L ambiguous?

$\Longleftrightarrow$ Is G Ambiguous

$\exists w \in L$

Mail: gatecse@appliedroots.com

s.t $\exists$ multiple derivations of w using G

Then G is ambiguous

G: $S \rightarrow a \mid aAb \mid abSb$

$A \rightarrow aAAb \mid bS$

w = abab

(1) $S \rightarrow abSb$
$\rightarrow abab$

(2) $S \rightarrow aAb$
$\rightarrow absb$
$\rightarrow abab$

(12) Complement problem:

Are both L and $\bar{L}$ of the same type?

are L & $\bar{L}$: reg?

are " : DCFL?

" " : CFL?

" " : REC?

" " : CSL?

" " : REL?

Decidability of problems about Languages:

Decision problem	Reg languages	DCFL	CFL	CSL	REC	REL
① Membership	D	D	D	D	D	UD
② Emptiness	D	D	D	UD	UD	UD
③ Finiteness	D	D	D	UD	UD	UD
④ Equivalence	D	UD	UD	UD	UD	UD
⑤ Totality/Completeness	D	D	UD	UD	UD	UD
⑥ Subset	D	UD	UD	UD	UD	UD
⑦ Intersection Finiteness	D	UD	UD	UD	UD	UD
⑧ Co-finiteness	D	D	UD	UD	UD	UD
⑨ Regularity	D	D	UD	UD	UD	UD
⑩ Ambiguity	D	UD	UD	UD	UD	UD
⑪ Complement	D	UD	UD	D	D	UD
⑫ Intersection Empty	D	UD	UD	UD	UD	UD

Solved problems on undecidability:

① Consider the following problems. L(G) denotes the language generated by a grammar G. L(M) denotes the language accepted by a machine M.

(I) For an unrestricted grammar G and string w, whether $w \in L(G)$ — UD

(II) Given a Turing Machine M, whether L(M) is Regular. — UD

(III) Given two grammars G_1 and G_2, whether $L(G_1) = L(G_2)$ — UD

(IV) Given a NFA N, whether there is a deterministic PDA P such that N and P accept the same language. — Decidable

NFA ⇓ RL; PDA P: DCFL

Which one of the following statements is correct?

(A) only I and II are undecidable ✗

(B) only II is undecidable ✗

(C) only II and IV are undecidable ✗

(D) only I, II and III are undecidable ✓

DCFL

RL

② Let A and B be finite alphabets and let # be a symbol outside both A and B. Let f be a total function from A^* to B^*. We say f is Computable if there exists a Turing machine M which given an input $x \in A^*$, always halts with f(x) on its tape. Let L_f denote the language

$\{x \# f(x) \mid x \in A^*\}$

UTM (halts)

which of the following statements is true?

A: f is Computable if and only if L_f is recursive.

B: f is Computable if and only if L_f is Recursively Enumerable ✗

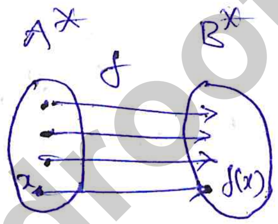
C: If f is Computable then L_f is recursive, but not conversely. ✗

D: If f is Computable, then L_f is recursively Enumerable, ✗ but not Conversely.

always Halts

$\Downarrow$

HALT



(3) let L(R) be the Language represented by regular Expression R. L(G) be the language generated by a Context free Grammar G. let L(M) be the language accepted by a Turing Machine M. which of the following decision problems are undecidable?

(I) Given a regular Expression R and a String w, is $w \in L(R)$? — D

(II) Given a Context-free Grammar G, $L(G) = \phi$? — D

(III) Given a Context-free Grammar G, $L(G) = \Sigma^*$ for some Alphabet Σ? — UD

Totality/ Completeness

(IV) Given a TM M and a String w, is $w \in L(M)$? — UD

× (a) I and IV only

× (b) II and III only

× (c) II, III and IV only

✓ (d) III and IV only.

(4) which of the following problems are undecidable?

✓ I. Given NFAs N_1 and N_2 is $\underbrace{L(N_1)}_{RL} \cap \underbrace{L(N_2)}_{RL} = \phi$? D

✓ II. Given a CFG $G = (N, \Sigma, P, S)$ and a string $x \in \Sigma^*$, does $\underbrace{x \in L(G)}_{D}$?

× III. Given CFG G_1 and G_2, is $L(G_1) = L(G_2)$? $\longrightarrow$ UD

× IV. Given a TM, M is $L(M) = \phi$? $\longrightarrow$ UD

M
E
F } CFL ↓ Decidable

(A) I and IV only

(B) I and III only

✓ (C) III and IV only

(D) II and IV only

(5) Consider the following statements

✓ (1) The complement of every Turing $\underbrace{\text{decidable}}_{\downarrow \text{REC}}$ language is Turing decidable.

NP
Algorithm
+halts
NTM
polynomial
Time

× (2) There exist some language which is in NP but not Turing decidable. ∃ TTM

✓ (3) If L is a language in NP, L is Turing decidable

which of the above statements is/are True?

(A) Only 2

(B) Only 3

(C) Only 1 and 2

(D) Only 1 and 3

1 - TRUE

2 - False

3 - True

Computational classes: P and NP, NPC, NP-Hard

TOC – Alg. problems ~ lang

Class-P: Polynomial : $n, n^2, n^3, n^4, n^5 \ldots n^k$

→ $\exists$ DTM that always halts. $\Leftrightarrow$ DTTM $\Leftrightarrow$ $\exists$ Det Alg

→ Input : n ; makes | halts within $O(T(n))$ moves | steps

Big-oh $T(n)$ = polynomial in n

DTM — Computer

step | Move $\longleftrightarrow$ basic operation

$q_i \xrightarrow{a} \bigcirc$

eg1 Is w in L(G)? G: CFG ← TOC

⇓

CYK algo → $O(n^3)$

eg2 Is $\exists$ a path vertex u to v in Graph

Dijkstra's → $O(n \log n)$

$= O(n^2)$

sorting : $O(n \log n)$

$O(n^2)$

Searching: $O(n)$

$O(\log n)$ $O(1)$

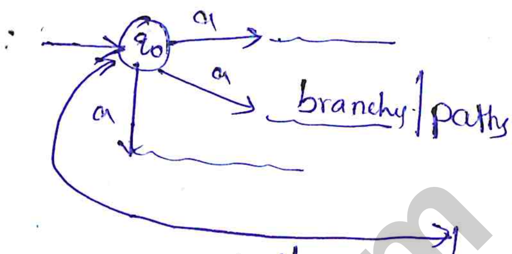
Class NP:

→ ∃ NTM that always halts ⟺ ∃ NTM

→ longest-path len is $O(T(n))$

$T(n)$: polynomial in n

transition-graph



executed | simulated parallely

DTM: single threaded program that halts.
↳ single path

NTM: Multi-threaded program that halts.
↳ OS

$P \subseteq NP \subset REC$

NTM ∼ DTM

~~Algo~~

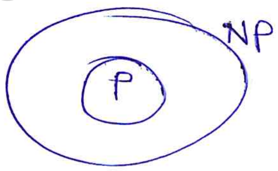
$P = NP$?
↳ Most important unsolved problems in cs/math.

Many beleive $P \neq NP$

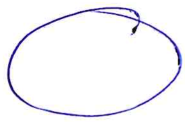
case 1: $P \neq NP$

case 2: $P = NP$

if $P \neq NP$



if $P = NP$



Thousands of known problems in NP for which no one could come up with a det-polynomial time.

NP Complete and NP Hard problems:

NPC, NPH {L-U → line}

Poly-time Reductions:

Instance of $P_1 \xrightarrow[\text{(P) poly-time DTTM}]{\text{reduction alg}}$ Instance of P_2.

P_1 & P_2 : TM, Graphs, arrays, sets, strings

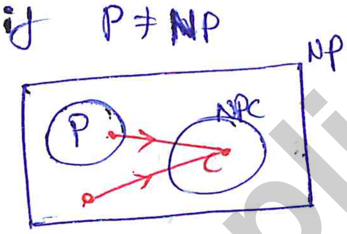
If P_2 is in P then ∃ DTTM to solve P_2

$n^2 \cdot n^3 = n^5$

$$\left\{ \begin{array}{l} \text{poly} \cdot \text{poly} = \text{poly} \\ \text{poly} + \text{poly} = \text{poly} \end{array} \right\}$$

NP-Complete: (NPC / NP-C)

if $P \neq NP$



P and NPC are disjoint sets

A problem C is in NPC if

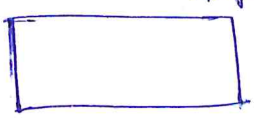
(i) $C \in NP$

(ii) Every problem in NP is reducible to C in poly-time.

$$\underset{\text{NP}}{\underset{\in}{\forall x}} \xrightarrow{\text{poly}} C \in NP$$

{ if P = NP

⇒ P = NP = NPC



NPC: $P \overset{?}{=} NP$

Today: We mostly believe $P \neq NP$

↳ we don't have a poly-time deti-Algorithm to any problem in NPC

↳ we do have exp time algorithm (2^n)

$O(n^n)$ $O(n!)$

If we show one problem in NPC to have a poly-time algorithm.

$\forall NP \xrightarrow{\text{polytime}} \underset{100\text{'s}}{\overset{\text{any}}{NPC}} \longrightarrow$ polynomial time Algorithm.

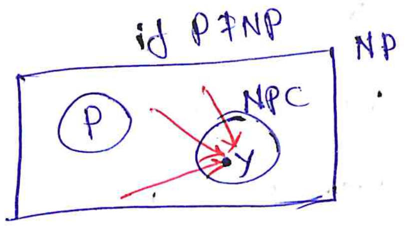
Then $\boxed{NP = P}$

How to prove a problem is NPC:

is $x \in NPC$? $<$ $x \in NP$

take $y \in NPC$

$y \xrightarrow{\text{poly}} x$



$\forall NP \xrightarrow{\text{poly}} y \xrightarrow{\text{poly}} x$

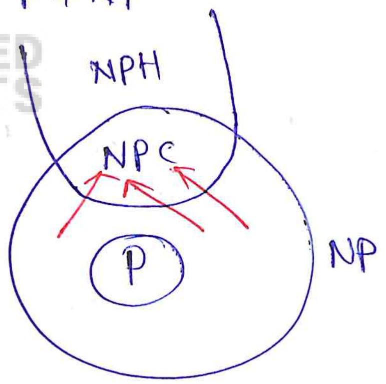
$\forall NP \xrightarrow{\text{poly+poly}} x$

NP-Hard:

{ x is in NPH if
∃ a polytime reduction from every problem in NP }

x need not belongs to NP.

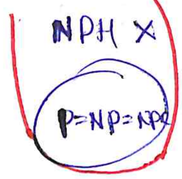
if $P \neq NP$



$NPC \subset NPH$

$NPH \,\&\, NP \Rightarrow NPC$

if $P = NP$



$P = NP = NPC$

Hard $\Rightarrow$ Non-polynomial time.

List of NP-Complete problems:

P: $O(n), O(1), O(\log n), O(n \log n)$

$O(n^2), O(n^2 \log n), O(n^3)$

① 1-planarity

② Graph coloring

③ Hamiltonian

④ Vertex Cover

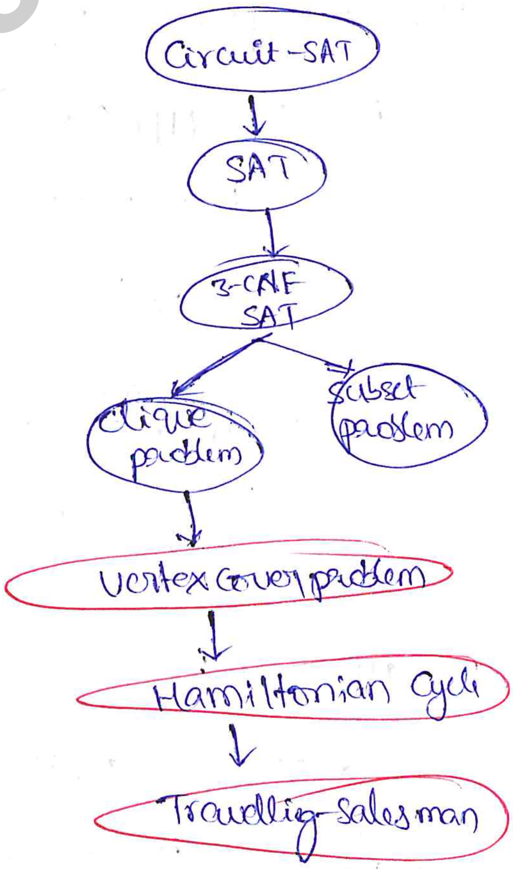
⑤ Knapsack problem

⑥ Subset Sum problem

⑦ Closest String

⑧ Longest common

⑨ Boolean Sat problem



NP-Hard :

1. Halting problem $\longrightarrow$ NPH, but not NP Complete
2. Subset sum problem

$\boxed{NPC \subset NPH}$

3. Travelling Salesman problem