INTRODUCTION OF TOC

- **Automata** :- Study of abstruct computing devices or machines.

- **Symbol** :- A symbol is an abstruct entity.
  Example: a, b, 0, 1, ...

- **Alphabet (Σ)** :- An Alphabet is a finite collection of Symbols.
  Example: $S = \{a, b, c\}$
  $\Sigma = \{0, 1\}$

- **String** :- It is a sequence of Symbols.
  Example :- $\Sigma = \{a, b\}$
  strings :- {a, b, aa, ab, ba, bb} hence have '6' string.
  ↳ length-2
  → Null or Empty string can be denoted by ($\in$ or Λ or λ).
  [If the length of string is zero, such string is called empty string]

- **Language** :- collection of string.
  [where strings is restricted over given alphabet]
  → $\Sigma = \{a, b\}$
  L_1 = set of all string of length 2.
  $= \{aa, ab, ba, bb\}$
  where language L_1 is finit language.

  → $\Sigma = \{a, b\}$
  L_2 = set of all string where each string starts with 'a'.
  $= \{a, aa, ab, aaa, aab, aba, ...\}$
  where, L_2 is infinit language.

- **length of string** :- No of symbols contained in a string.
  $|\{ab\}| \Rightarrow$ length '2'. $|\in| \Rightarrow$ length '0'.

- **Prefix of a String :-** If $W = XY$, for some string Y, then Y is a prefix of W.

  example- $W = \{0011\}$

  $\Rightarrow \{\underset{X}{001}\ \underset{Y}{1}\} \Rightarrow \{\underset{X}{00}\ \underset{Y}{11}\} \Rightarrow \{0, 00, 001\}$

- **Suffix of a string :-** If $W = XY$, for some string X, then Y is a suffix of W.

  example- $W = \{0011\}$

  $= \{\underset{X}{001}\ \underset{Y}{(1)}\} \Rightarrow \{1, 11, 011\}$

  $\rightarrow$ suffix.

- **Kleen closure :-**

$\rightarrow$ It is denoted by *(asterisk) after the name of the alphabet is Σ^*. This notation also known as the Kleene Star.

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

Σ^0 = set of all strings of length '0' $= \{\epsilon\}$.

Σ^1 = set of all strings of length '1' $= \{a, b\}$

$\Sigma^2 = \Sigma . \Sigma = \{a, b\}\ \{a, b\}$

$= \{aa, ab, ba, bb\}$ = set of all strings of length '2'.

$\Sigma^3 = \Sigma\ \Sigma\ \Sigma =$ set of all strings of length '3'.

$|\Sigma|^3 = 8$ (no of string)

$\vdots$

$|\Sigma|^n = 2^n$ = set of all strings of length 'n'.

$\therefore \boxed{\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots\ldots}$

$= \{\epsilon\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \cup \ldots\ldots\ldots$

Σ^* = set of all string possible over $\{a, b\}$ [Universal set]

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

$\Sigma^* = \boxed{L_1 \quad L_2 \quad L_3}$

$\begin{bmatrix} L_1 \subseteq \Sigma^* \\ L_2 \subseteq \Sigma^* \\ L_3 \subseteq \Sigma^* \end{bmatrix}$

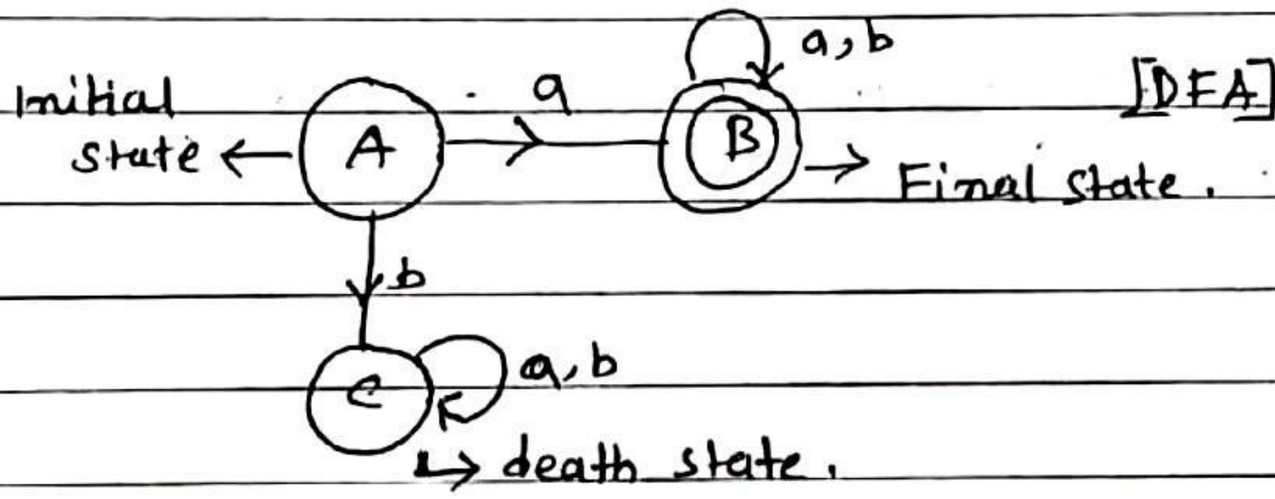
→ no of string possible over Σ^* is infinite.
→ no of Language possible over Σ^* is infinite.

L (Language)

(String) S → FA → Yes,
→ No.

→ A program is a string.
→ C-programming language = set of all valid program.

L = set of all strings which starts with 'a'.
L = { a, aa, ab, aaa, ... }



[DFA]

String = aab (present in language or not)
this string are accepted by FA. $A \xrightarrow{a} B \xrightarrow{a} B \xrightarrow{b} (B)$

→ the string will be accepted by FA, if after scaning entire string we reach in final state from initial state.

String = bba
this string are not accepted by FA. $A \xrightarrow{b} C \xrightarrow{b} C \xrightarrow{a} C$

- Positive Closure:
→ The '+' (plus operation) is some times called positive closure.
→ If $\Sigma = \{a, b\}$, then.
$\Sigma^+ = \{ a, aa, ab, ba, bb, \ldots \}$
$\boxed{\Sigma^+ = \Sigma^* - \{\epsilon\}}$

- **Substring of a string :-**

→ A string 'w' = abc

here, ab, bc are substring of w.

But ac are not substring of w.

- **Concatenation of two string :-**

→ If $x, y \in \Sigma^*$, then x concatenated with y is the word formed by the symbol of 'x' followed by the symbols of 'y'.

→ This is denoted by $x \cdot y$, It is same as xy.

- **Reversal of a string :**

→ Given a string w, it reversal denoted by w^R is the string spelled back words.

$w = ab$

$w^R = ba$.

- **Grammar :**

→ It enumerates string of the language.

→ It is a finite set of rules defining a language.

→ A grammar is defined as 4-tuples (V, T, P, S)

Where,
- V → Set of non terminals.
- T(Σ) → Set of input terminals.
- P → finite set of production rule.
- S → start of symbol.
  → start(S).

example :-

(production) P =
- S → aSB
- S → aB
- B → b

here, V = {S, B}

T = {a, b}

→ Getting a string from a grammar is called Derivation.

Derivation on aabb

$S \rightarrow aSB$

$\rightarrow aaBB \quad [S \rightarrow aB]$

$\rightarrow aabB \quad [B \rightarrow b]$

$\Rightarrow \boxed{aabb} \quad [B \rightarrow b]$

→ entire process called derivation.

→ sequential forms.

Q-1 Construct a grammar given the following language,

$L = \{$ Set of all string length '2' $\}$

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

→ $L = \{aa, ab, ba, bb\}$ $\quad \underbrace{(a+b)}_{A} \underbrace{(a+b)}_{A}$

$S \rightarrow aa/ab/ba/bb.$

$S \rightarrow AA$

$A \rightarrow a/b$

→ production rules.

Q-2 construct grammar, given the following language,

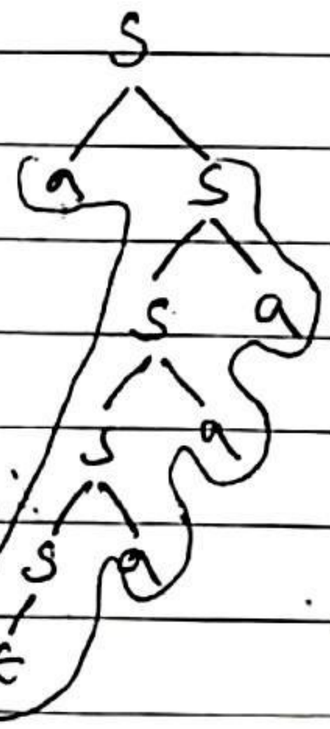
$L = \{a^n / n \geq 0\}$

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

P.R →

$S \rightarrow aS/\epsilon$

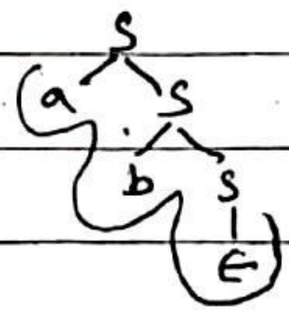
$S \rightarrow Sa/\epsilon$



(a^4).

Q-3 Construct grammar to generate $(a+b)^*$.

→ $\boxed{S \rightarrow aS/bS/\epsilon}$ ∴



(ab)

Q-4 Construct a grammar, given the following language,

$L = \{$ set of all string of length atleast '2' $\}$

→ $L = \{aa, ab, ba, bb, aaa, aab \cdots\cdots\}$

$\underbrace{(a+b)}_{A}\ \underbrace{(a+b)}_{A}\ (a+b)^*$

production rules,

$S \rightarrow AAB$
$A \rightarrow a/b.$
$B \rightarrow aB/bB/\epsilon.$

Q-5 C.A.G, given the following language,

$L = \{$ string of length at most '2' $\}$.

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

$\underbrace{(a+b+\epsilon)}_{A}\ (a+b\epsilon)$

production rules;

$S \rightarrow AA$

$A \rightarrow a/b/\epsilon$

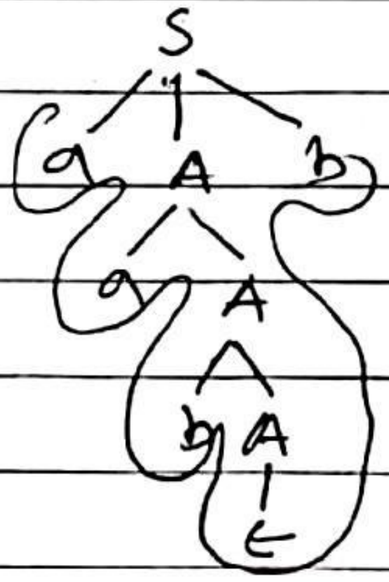
Q-6 $L = \{$ start with 'a' and end with 'b' $\}$.

→ $a\ (a+b)^*\ b$

production rules,

$S \rightarrow aAb$

$A \rightarrow aA/bA/\epsilon$



aabb

Q-7 $L = \{$ all string starting and ending with different symbol $\}$.

→ $a(a+b)^*b + b(a+b)^*a$

PR:- $S \rightarrow aAb/bAa.$

$A \rightarrow aA/bA/\epsilon$

Q-8 $L = \{$ set of all string starting and ending with same symbol $\}$

→ $a\,(a+b)^*\,a + b\,(a+b)^*\,b.$

$S \rightarrow aAa / bAb / b / a / \epsilon$

$A \rightarrow aA / bA / \epsilon$

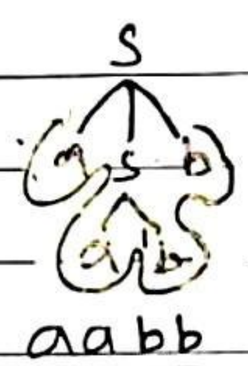
Q-9 construct a grammar, given the following language.

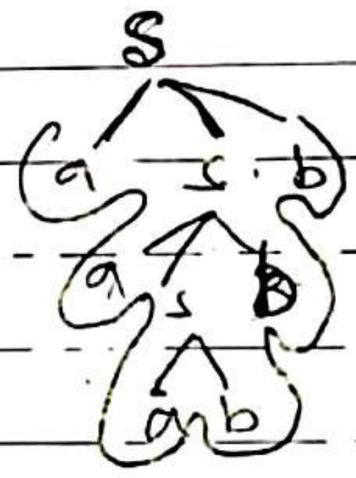
$L = \{a^n b^n / n \geq 1\}$

→ $L = \{a^n b^n / n \geq 1\}$

$= \{ab, aabb, \ldots\}$

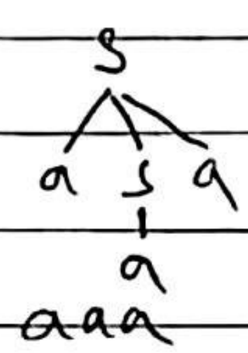
$S \rightarrow aSb / ab / \epsilon$

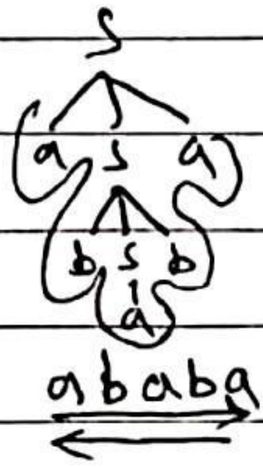




Q-10 construct grammars that generate set of all palindrom - WW^R, WaW^R, WbW^R

→ $S \rightarrow aSa / bSb / a / b / \epsilon$





Q-11 Construct grammar that generate "even length string".

→ $L = \{aa, ab, ba, bb, aaaa, \ldots\}$

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

$S \rightarrow BS / \epsilon$

$B \rightarrow AA$

$A \rightarrow a / b$

Q-12 $L = \{a^n b^m / n, m \geq 1\}$

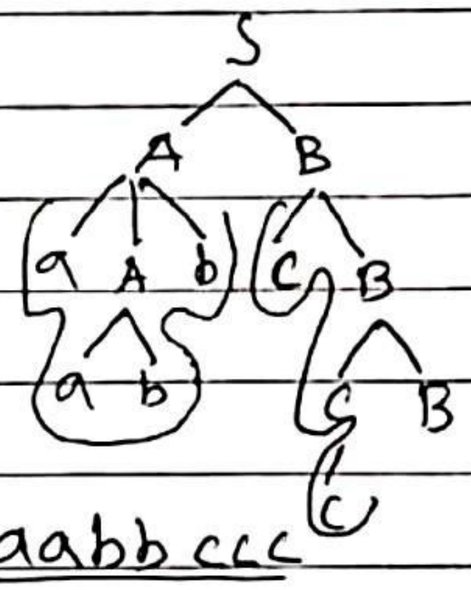
→ $L = \{ab, aab, aabb, abb, \cdots\}$

$S \rightarrow AB$	
$A \rightarrow aA / a$	$\rightarrow a^n$
$B \rightarrow bB / b$	$\rightarrow a^m$

Q-13 $L = \{a^n b^n c^m / n, m \geq 1\}$

→

$S \rightarrow AB$
$A \rightarrow aAb / ab$
$B \rightarrow cB / c$



aabbccc

Q-14 $L = \{a^n c^m b^n / n, m \geq 1\}$

→ $S \rightarrow aSb / aAb$
$A \rightarrow cA / c$

aacbb

Q-15 $L = \{a^n b^n c^m d^m / n, m \geq 1\}$

→

$S \rightarrow AB$	
$A \rightarrow aAb / ab$	$\rightarrow a^n b^n$
$B \rightarrow cBd / cd$	$\rightarrow c^m d^m$

Q-16 $L = \{a^n b^{2n} / n \geq 1\}$

→ $S \rightarrow aSbb / abb$

Q-17 $L=\{a^{m+n}\ b^m\ c^n \mid m,n \geq 1\}$ $\{a^n b^m c^n d^m \mid n,m \geq 1\}$ (×)

$\rightarrow$ $L=\{a^m a^m b^m c^n \mid m,n \geq 1\}$

$S \rightarrow aSc / aaAc$.
$A \rightarrow aAb / ab$.

Q-18 $L=\{a^n b^{n+m} c^m \mid n,m \geq 1\}$

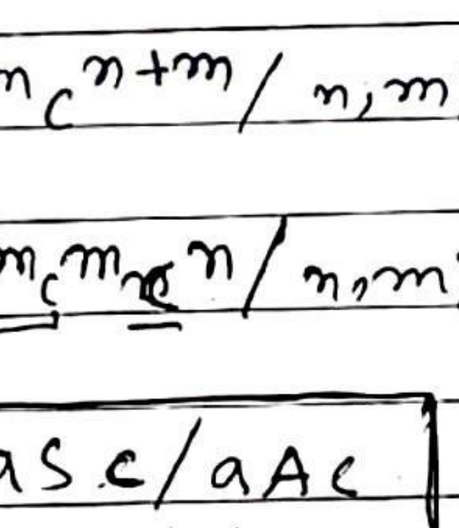
$\rightarrow$ $L=\{a^n b^n b^m c^m \mid n,m \geq 1\}$

$S \rightarrow AB$
$A \rightarrow aAb / ab$
$B \rightarrow bBc / bc$

Q-19 $L=\{a^m b^m c^{n+m} \mid n,m \geq 1\}$

$\rightarrow$ $L=\{a^m b^m c^m c^n \mid n,m \geq 1\}$

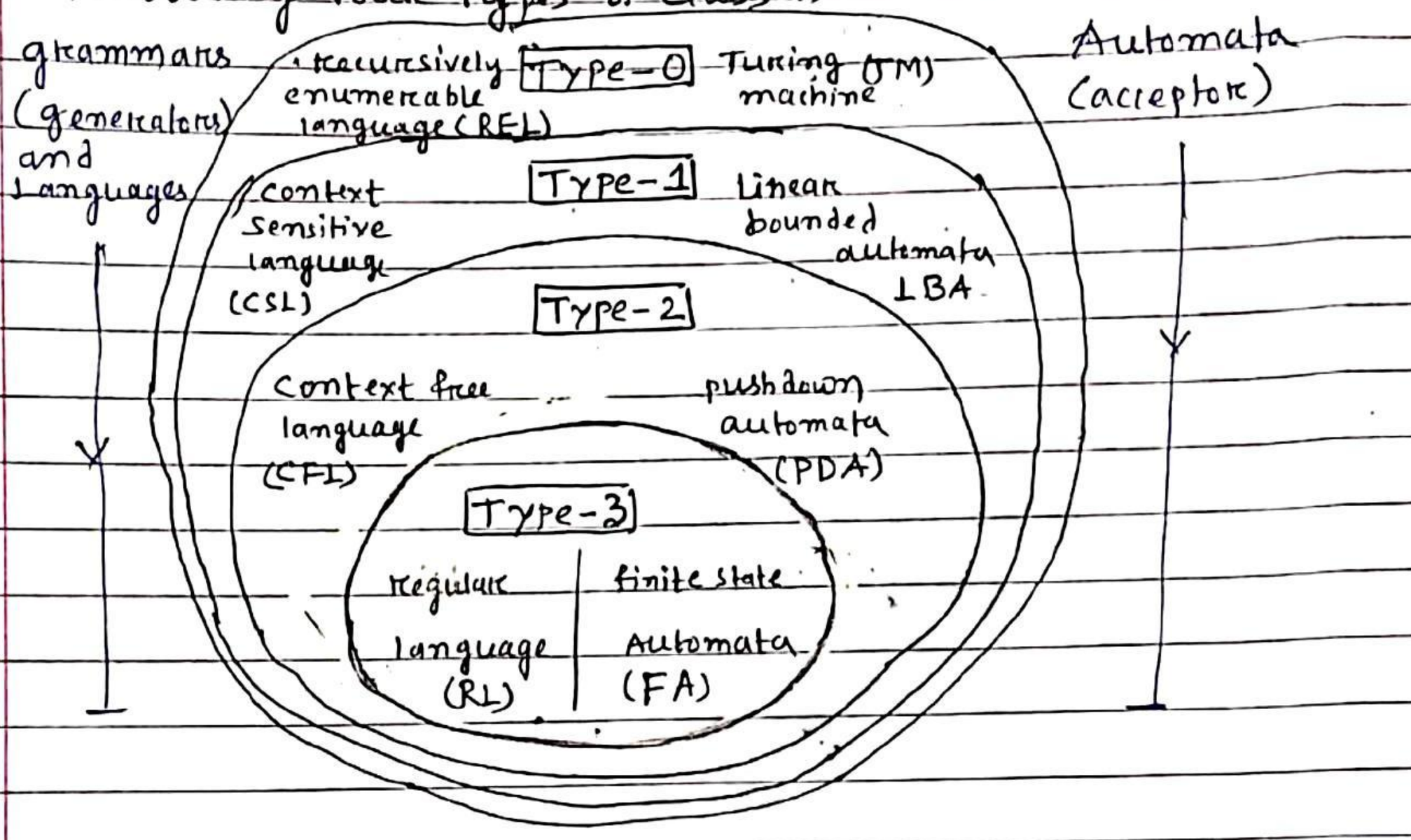
$S \rightarrow aSc / aAc$
$A \rightarrow bAc / bc$



$aabbcccc \Rightarrow a^{(m)}\ a^{(m)}\ b^2\ c^{(2+2)}$

$\Rightarrow a^2 b^2 c^4$. ✓

- Comsky Hierachy: Comsky hierachy consists of following four types of classas-



(Chomsky Hierarchy)

- Types of Grammers :-

① TYPE-0 Grammar (Unrestricted Grammar):

→ These are unrestricted grammar which include all formal language.

→ This grammar generate exactly all language that can be recognized by a turing machine.

→ Rules are of the form $\alpha \rightarrow \beta$

(α must have at least 1 non-terminal.)

→ where, α and β are are arbitrary sequence of terminals and non-terminals and $\alpha \neq \epsilon$.

ex- ✓ $A \rightarrow aA \mid bA$ | ✗ $a \rightarrow Ba$

(2) TYPE-1 Grammar (Context Sensitive Grammar):

→ language defined by type-1 grammars are accepted by the linear bounded automata.

→ Rules are of the form, $|\alpha| \leq |\beta|$
length of 'α' is less than or equal to length of 'β'.

$A \to \epsilon$ is not allowed unless A is a start symbol.

✗ $AB \to a$, ✓ $A \to aB$

(3) TYPE-2 Grammar (Context ~~Sensitive~~ free Grammar)

→ language defined by type-2 grammars are accepted by push down automata.

→ Rules are of the form $\alpha \to \beta$.
Where, α = single nonterminal.

$A \to a$ ✓ $Aa \to bB$ ✗.

(4) TYPE-3 Grammar (Regular Grammar)

→ language define by type-3 grammars are accepted by finite state automata.

→ Regular grammar can follow either right linear or left linear.

→ (right linear grammar)

$A \to \alpha B / \beta$

$A, B \in V$ (non terminal)

$\alpha, \beta \in T^*$ (terminal)

example,
$A \Rightarrow aB$
$B \to aB/bB/a/b$

→ left linear grammar

$A \to B\alpha / \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

example,
$A \to Ba$
$B \to Ba/Bb/a/b$

If, $\begin{bmatrix} A \to Ba/a \\ B \to aB/a \end{bmatrix}$ → not Type 3 grammar.

- Chomsky hircarchy Examples-
  - Identify grammar :-

1) Example :

$S \rightarrow aSb \mid aA \mid B$

$B \rightarrow aA \mid a$

$C \rightarrow <D \mid D$

→ ans is (Type-0)

It will be accepted by Turing machine.

2) Example

$S \rightarrow a \mid aA \mid Bb$

→ (Type-2).

It will be accepted by Push Down automata.

3) Example -

$S \rightarrow aSb \mid ab$ → (Type-2) grammar.

- Type-0 class is also called as :
  - → Unrestricted Grammar.
  - → Recursively Enumerable languages.
  - → Turing Machine.
- Type-1 class is also called as :
  - → Context Sensitive grammar.
  - → Context Sensitive language.
  - → Linear Bounded Automata.
- Type-2 class is also called as :
  - → Context Free Grammar.
  - → Context Free language.
  - → Push down automata.
- Type-3 class is also called as :
  - → Regular Grammar.
  - → Regular language.
  - → finit automata.

- Finite Automata (FA):
  → Machines with fixed amount of unstructured memory, accepts regular language.
  → Application of FA: useful for modeling chips, communication protocols, adventure games, some control system, etc.

- push down Automata (PDA):
  → Finite Automata with unbounded structured memory in the form of a push down stack, accepts CFL.
  → Application of PDA: ~~Compilers~~ useful for modeling parsing, compilers, programming language design.

- Turing Machine (TM):
  → Finite automata with unbounded tape accepts or enumerates recursively enumerable language.

  → equivalent to RAM's and various programming language models.
  → Application of TM: Model for general sequential computation (real computer).