SNS COLLEGE OF TECHNOLOGY

Coimbatore-36.

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A+’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE CODE& NAME : *19CSB301* &AUTOMATA THEORY AND COMPILER DESIGN

III YEAR/ V SEMESTER

UNIT – I FINITE AUTOMATA AND REGULAR LANGUAGES

Topic: Central concepts of Automata Theory

Dr.B.Vinodhini

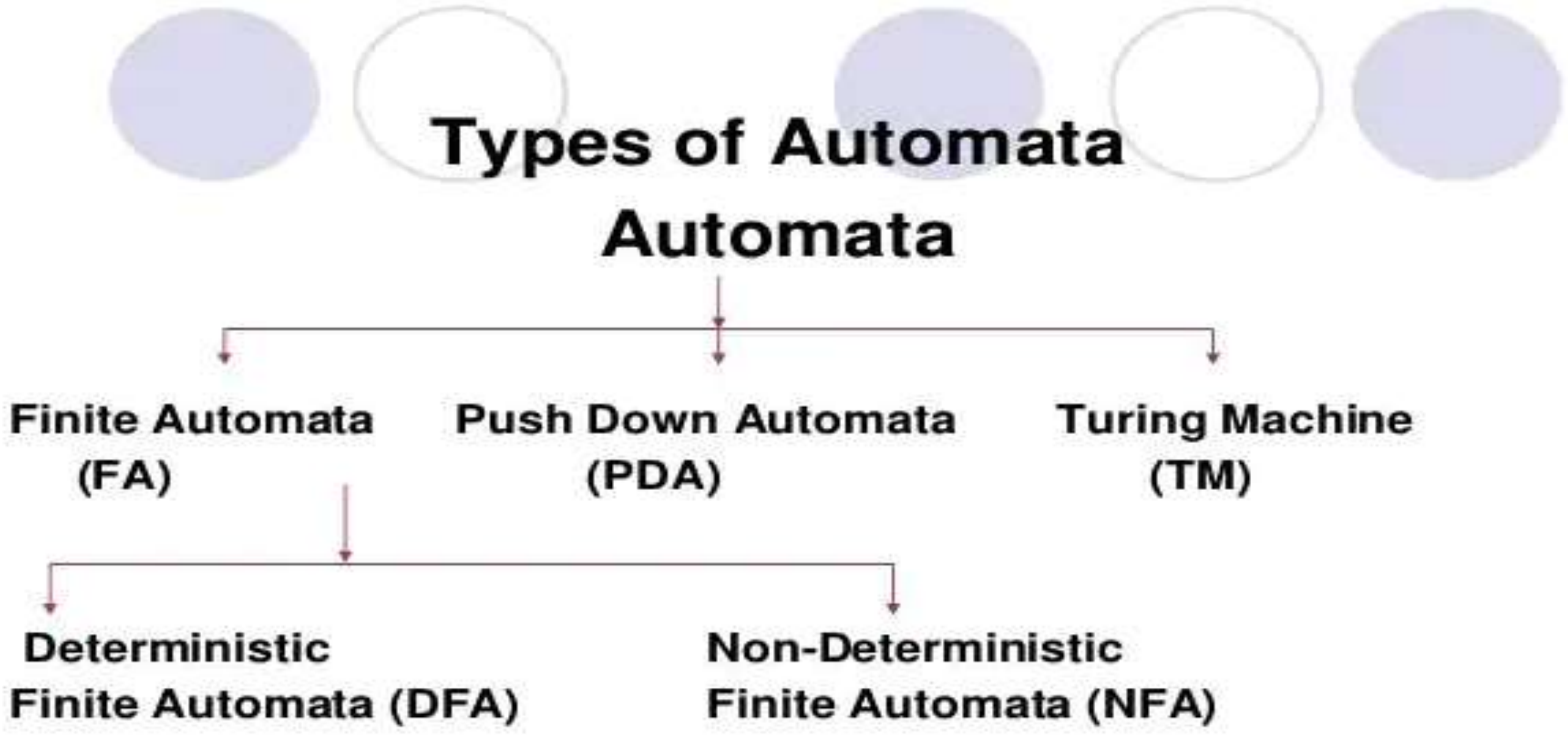
Assistant Professor

Department of Computer Science and Engineering

Introduction to Automata

- Theory of automata is a theoretical branch of computer science and mathematical.
- It is the study of abstract machines and the computation problems that can be solved using these machines.
- The abstract machine is called the automata.
- The main motivation behind developing the automata theory was to develop methods to describe and analyse the dynamic behaviour of discrete systems.
- This automaton consists of states and transitions. The **State** is represented by **circles**, and the **Transitions** is represented by **arrows**.
- Automata is the kind of machine which takes some string as input and this input goes through a finite number of states and may enter in the final state.

Types of Automata

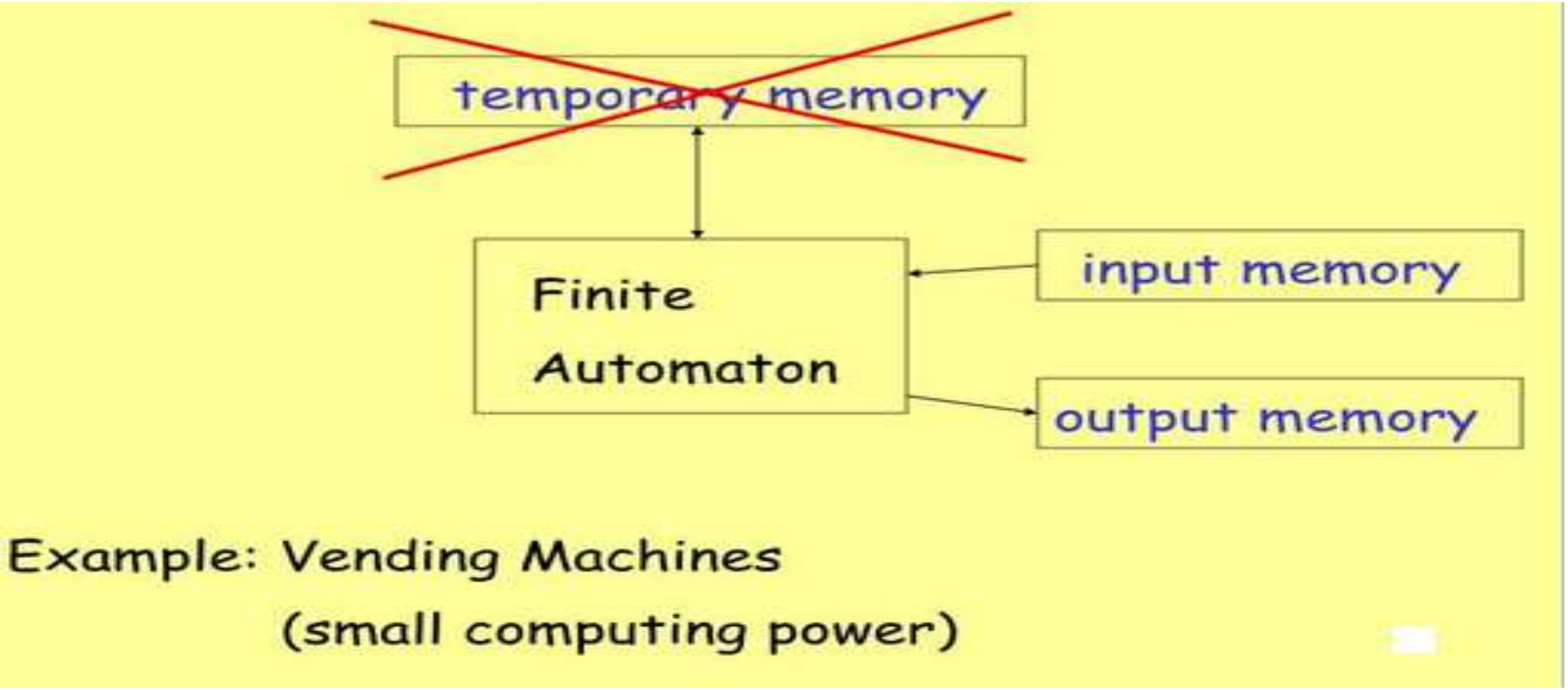


Types of Automata

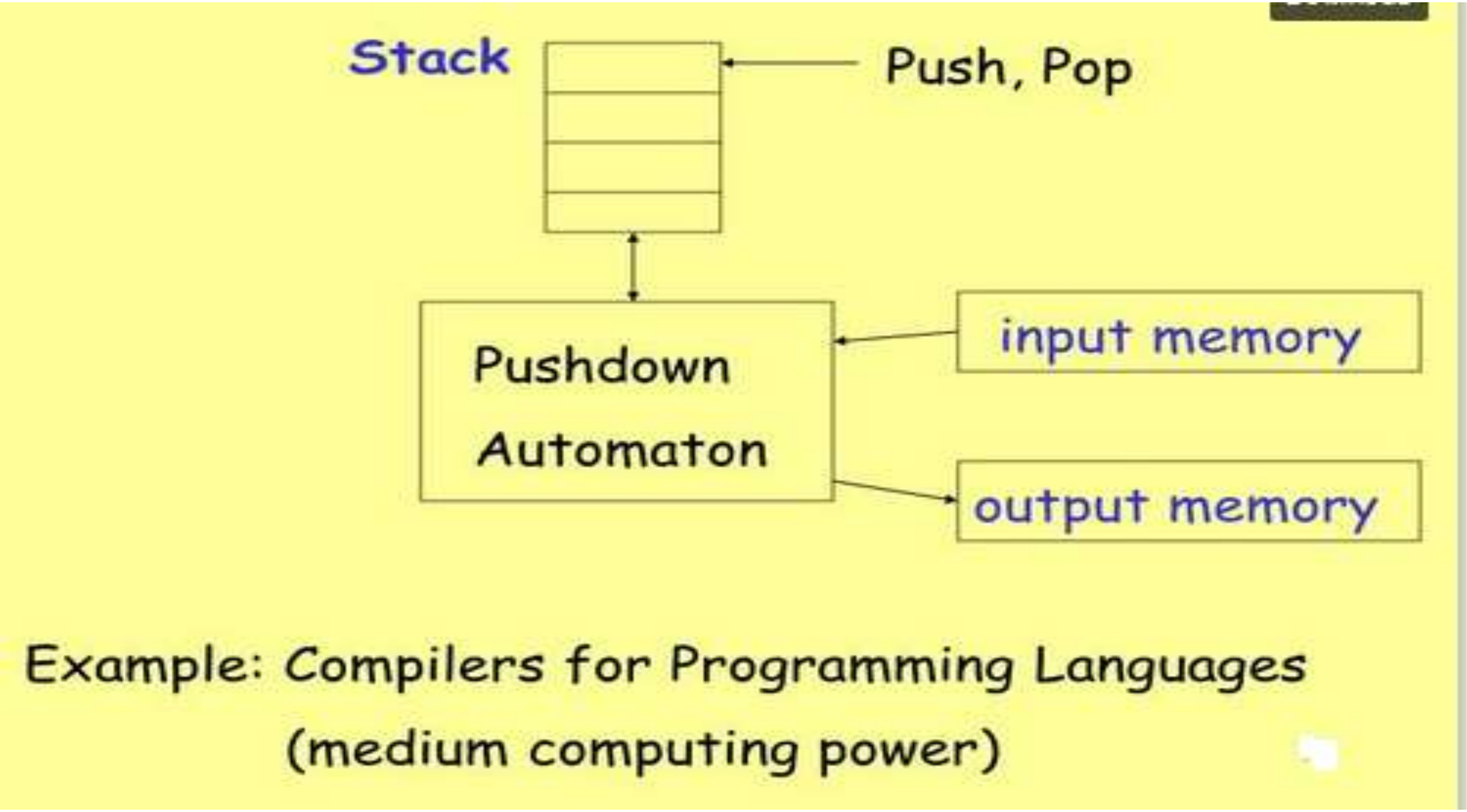
Automata are distinguished by the temporary memory

- **Finite Automata:** no temporary memory
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory

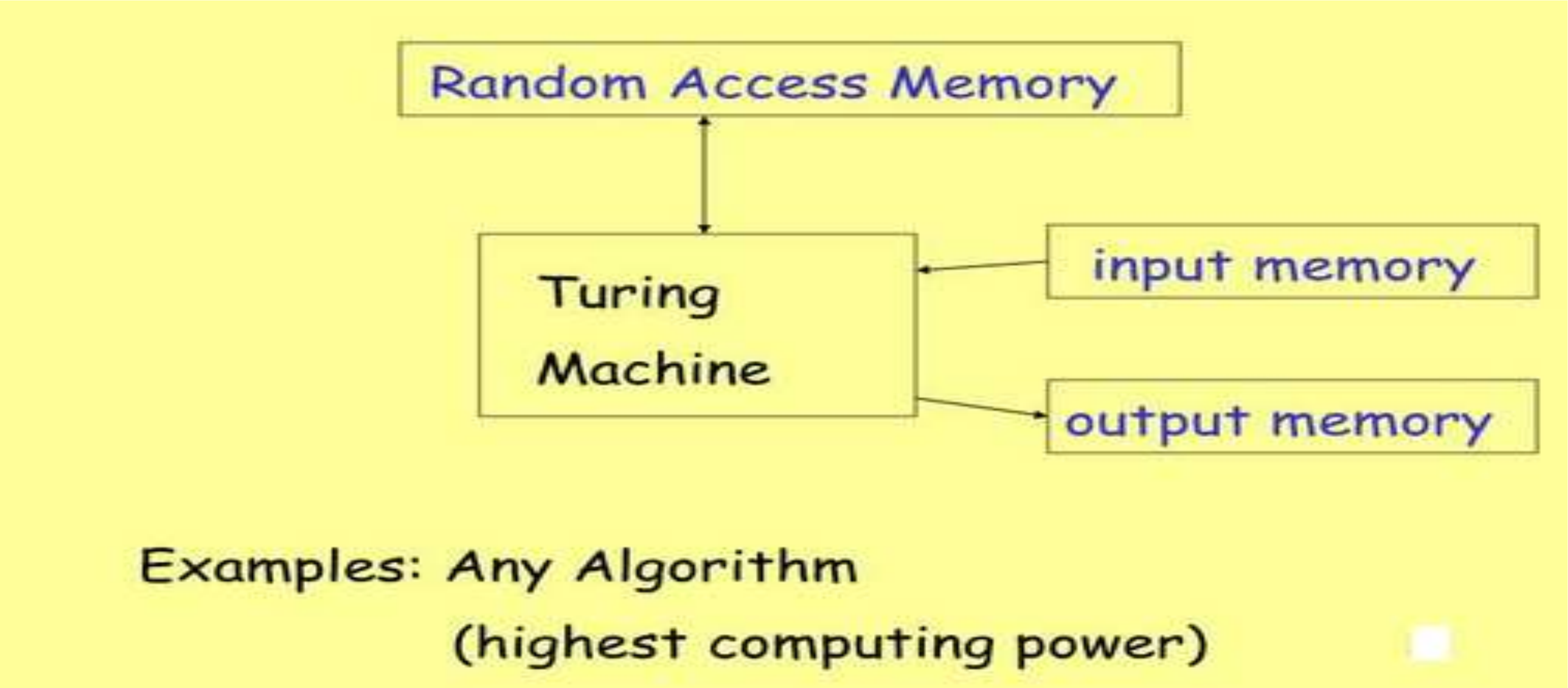
Types of Automata



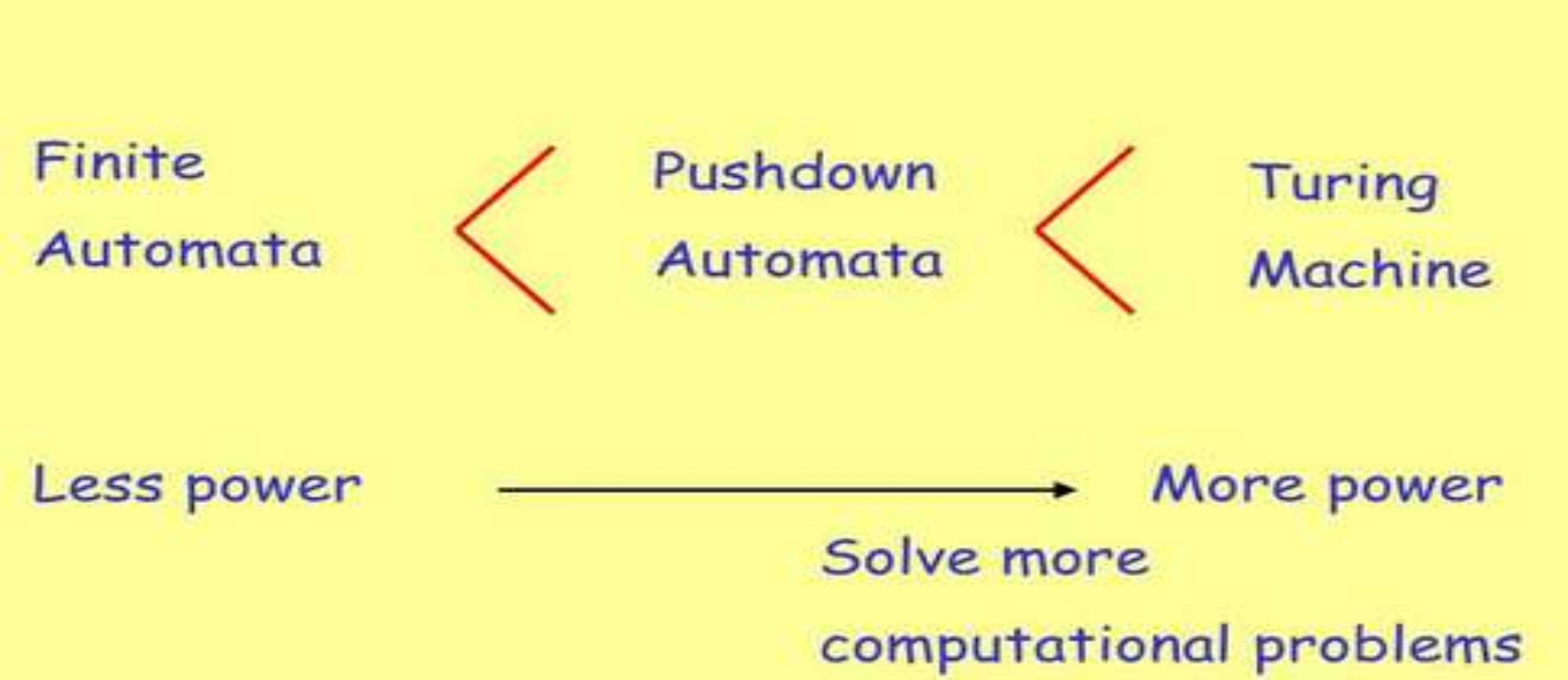
Types of Automata



Types of Automata



Types of Automata



Types of Automata

Model	Language Recognition	Memory Management	Implementation
Finite Automata	Regular Languages	No temporary memory	Elevators, Vending Machines, Traffic Light, Neural Network (small computing power)
Pushdown Automata	Context-free Languages	Stack	Compilers for Programming Languages (medium computing power)
Turing machine	Unrestricted Grammar, Lambda Calculus (Computable Languages)	Random access memory	Any Algorithm (highest computing power)

Key Terminologies

Symbols:

Symbols are an entity or individual objects, which can be any letter, alphabet or any picture.

Example:

1, a, b, #

Alphabets:

Alphabets are a finite set of symbols. It is denoted by ∑.

Examples

∑ = {a, b}   ∑ = {A, B, C, D}   ∑ = {0, 1, 2}   ∑ = {0, 1, ..., 5]

Key Terminologies

String:

It is a finite collection of symbols from the alphabet. The string is denoted by w.

Example 1:

If ∑ = {a, b}, various string that can be generated from ∑ are {ab, aa, aaa, bb, bbb, ba, aba.....}.

A string with zero occurrences of symbols is known as an empty string. It is represented by ε.

The number of symbols in a string w is called the length of a string. It is denoted by |w|.

Key Terminologies

Language(Set of Strings with Rules)

A language is a collection of appropriate string. A language which is formed over Σ can be **Finite** or **Infinite**.

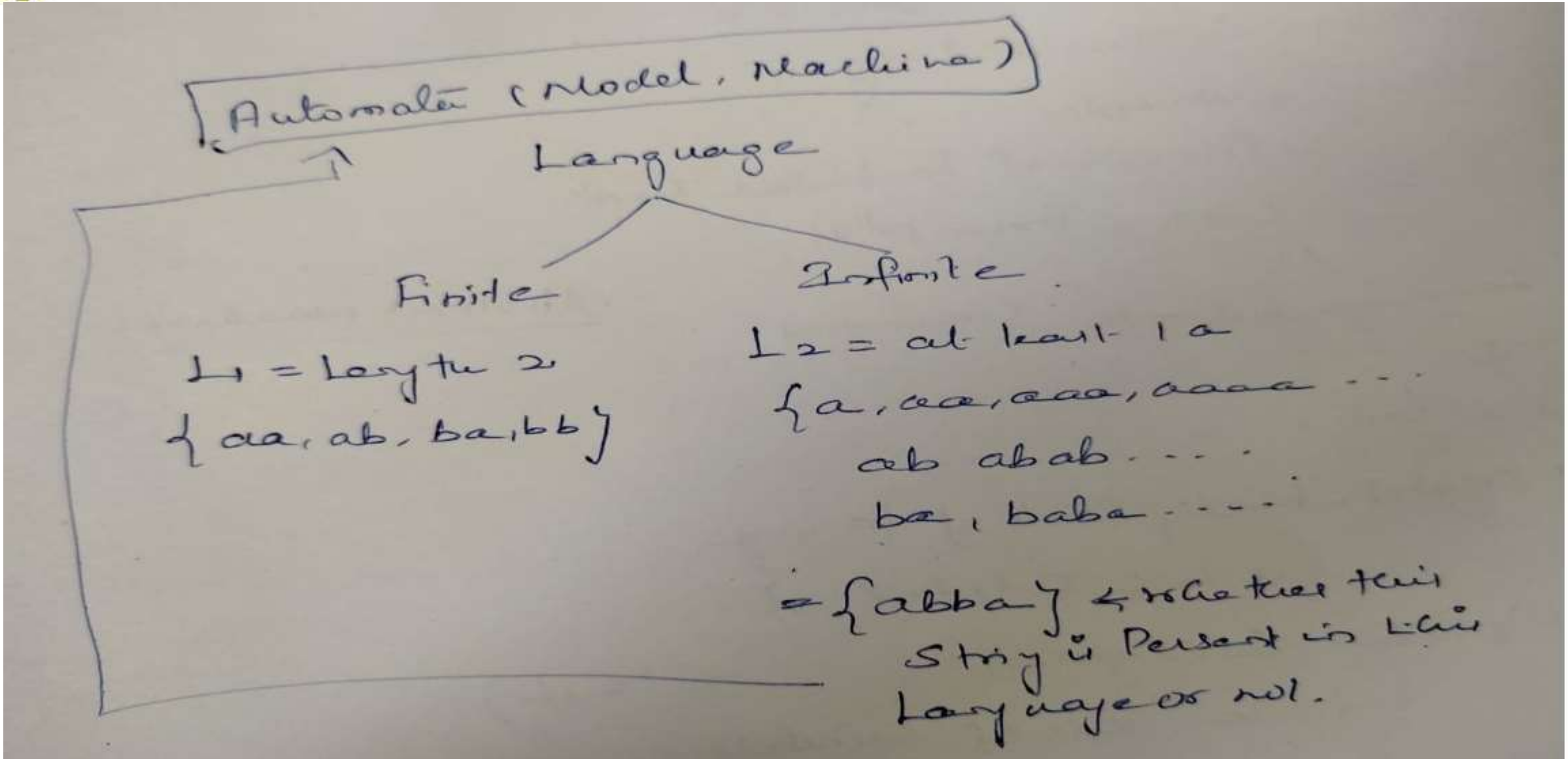
Example: 1

L1 = {Set of string of length 2} = {aa, bb, ba, bb} **Finite Language**

Example: 2

L2 = {Set of all strings starts with 'a'} = {a, aa, aaa, abb, abbb, ababb} **Infinite Language**

Key Terminologies



Key Terminologies

Empty String, Length of String, reverse of String, Power of alphabet, Power of String

$\Sigma = \{a,b\}$ ∧ a b aa ab

Empty String — string that has no letter, also known as **Null string**, denoted by ∧, λ or ε

It's length is Zero (0)

Length of String — is the number of letters in a string, denoted by **|s|**

Example: s = abab **|s| = 4** or **length(s) = 4** or **length(abab) = 4**

Reverse of String — Is obtained by writing letters of string in reverse order, denoted by Rev(s) or $\overset{r}{S}$ Or Reverse(s)

Example: s = abab **Rev(s)** = baba **Reverse(s)** = baba

Power of Alphabet — Determines that the strings made from alphabet will be of length equal to power of alphabet

$\Sigma = \{a,b\}^2$ {aa, ab, ba, bb}

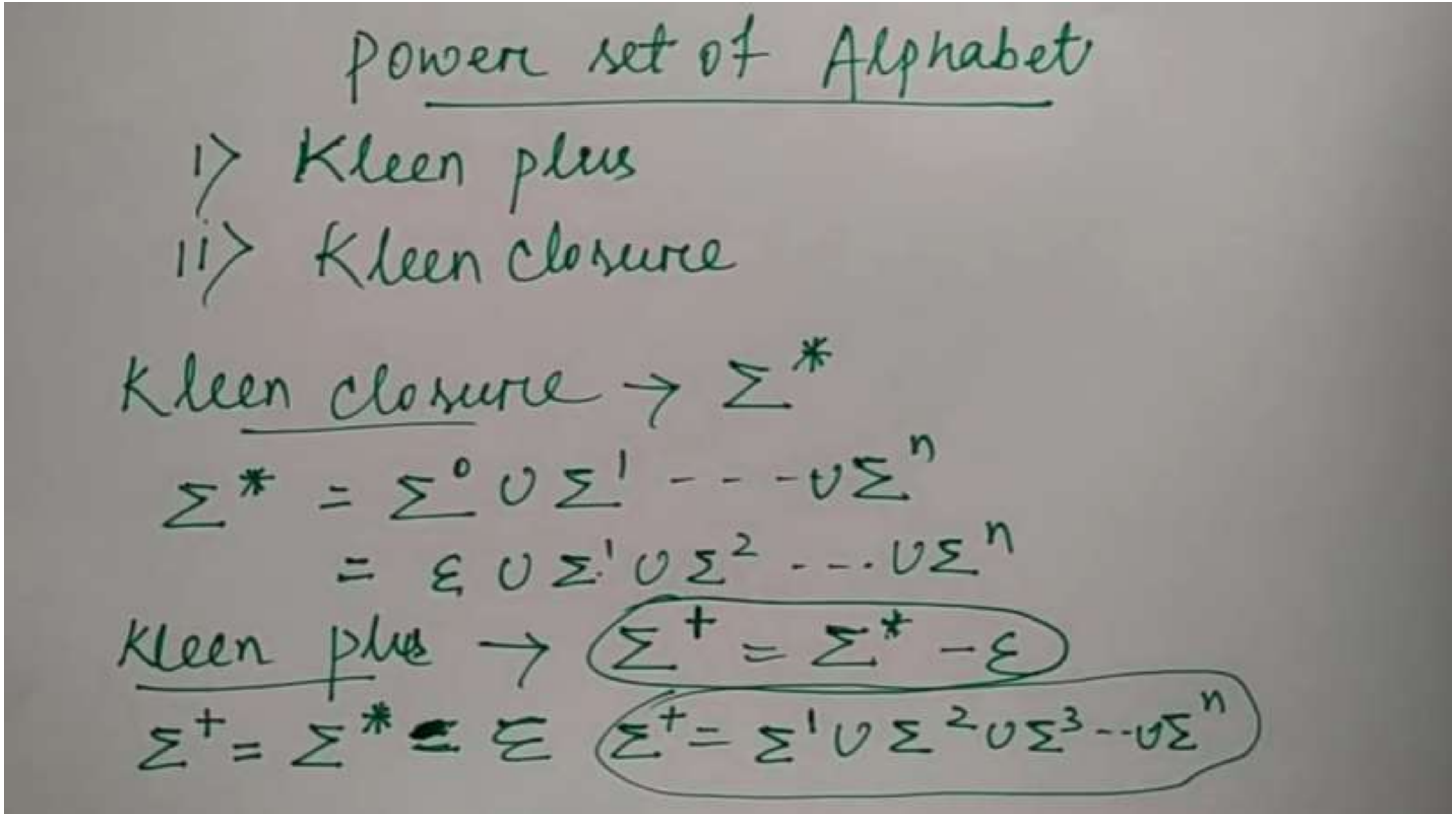
Total number of letters in alphabet → n^m (Length/power → m) $2^2 = 4$

Power of string — Determines the length of string

$(bab)^2$ = babbab

ba^2b = baab

Power of Alphabet



Power of Alphabet

Kleene Star/ Closure/Operator VS Kleener Closure/Plus/Positive , Lexicographic Order

Kleene Star
Kleene Closure
Kleene Operator

It is undermined power, represent infinite number of terms can be made including empty string

Denoted by *

Kleene Plus
Kleene Positive
Positive Closure

It is undermined power, represent infinite number of terms can be made except empty string

Denoted by +

Lexicographic order Method of Sequencing a language in which strings are grouped by their length (i.e. strings of shortest length first)

Power of Alphabet Determines that the strings made from alphabet will be of length equal to power of alphabet

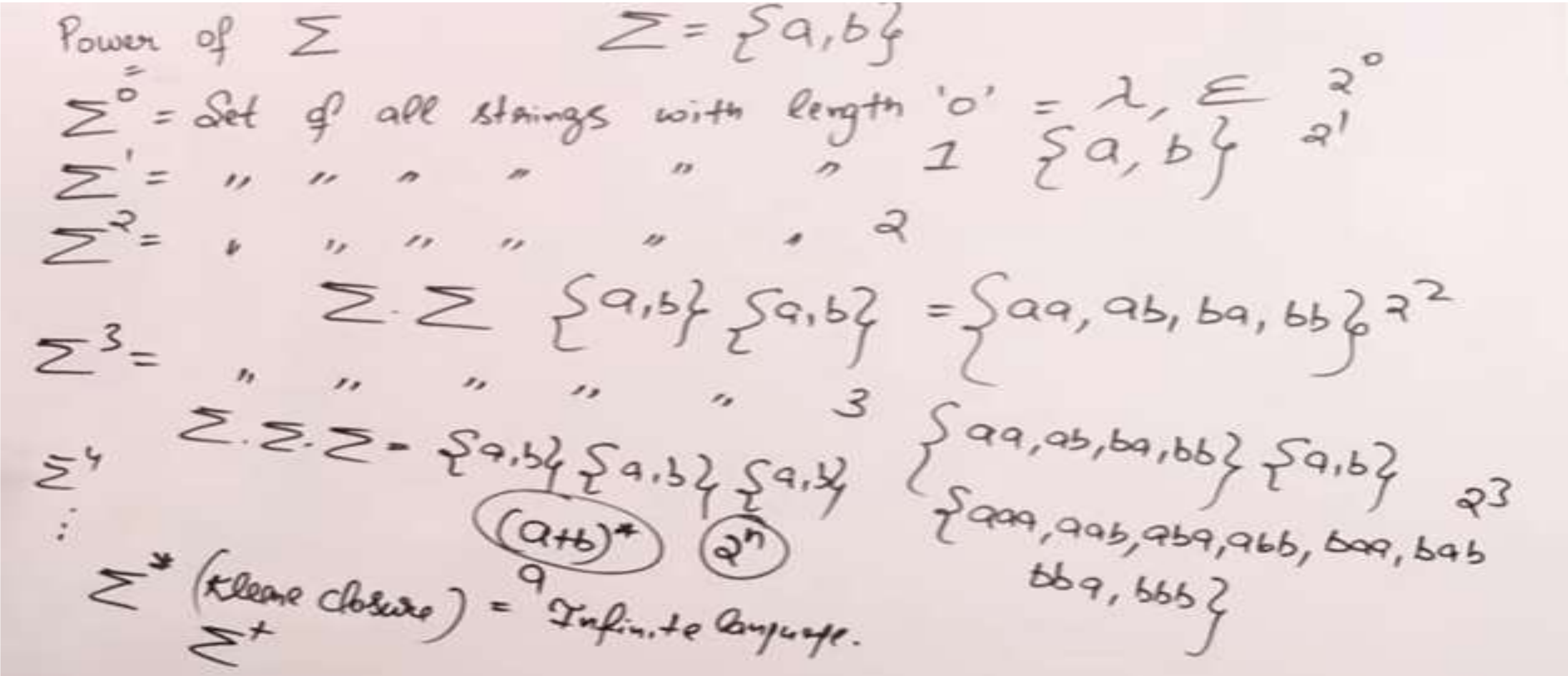
$\Sigma = \{a,b\}$

$\Sigma^2 = \{a,b\}^2$	Σ^2	{aa, ab, ba, bb}
$\Sigma^* = \{a,b\}^*$	Σ^*	{ ^ , a, b, aa, ab, ba, bb, aaa, aab, ... }
$\Sigma^+ = \{a,b\}^+$	Σ^+	{ a, b, aa, ab, ba, bb, aaa, aab, ... }

Power of string Determines the length of string

ba^2b = baab
ba^*b = bb or bab or baab or baaaaaab
ba^+b = bab or baab or baaaaaab

Power of Alphabet



Types of Grammar

Types of Grammar

- *Grammar in Automata*
  - **G = (V, T, P, S)**
  - V – Non-Terminals / Variables /Auxillary Symbols (A,B,C,….)
    - Takes part in generation of sentence (Not a part of sentence)
  - T – Terminals (small-case letters a,b,c,….)
  - P – Production Rules
  - S – Start Symbol

Example1

V = {S}

T = {a, b}

P = {S→aSbS, S→bSaS, S→ ∈}

S = {S}

Example2

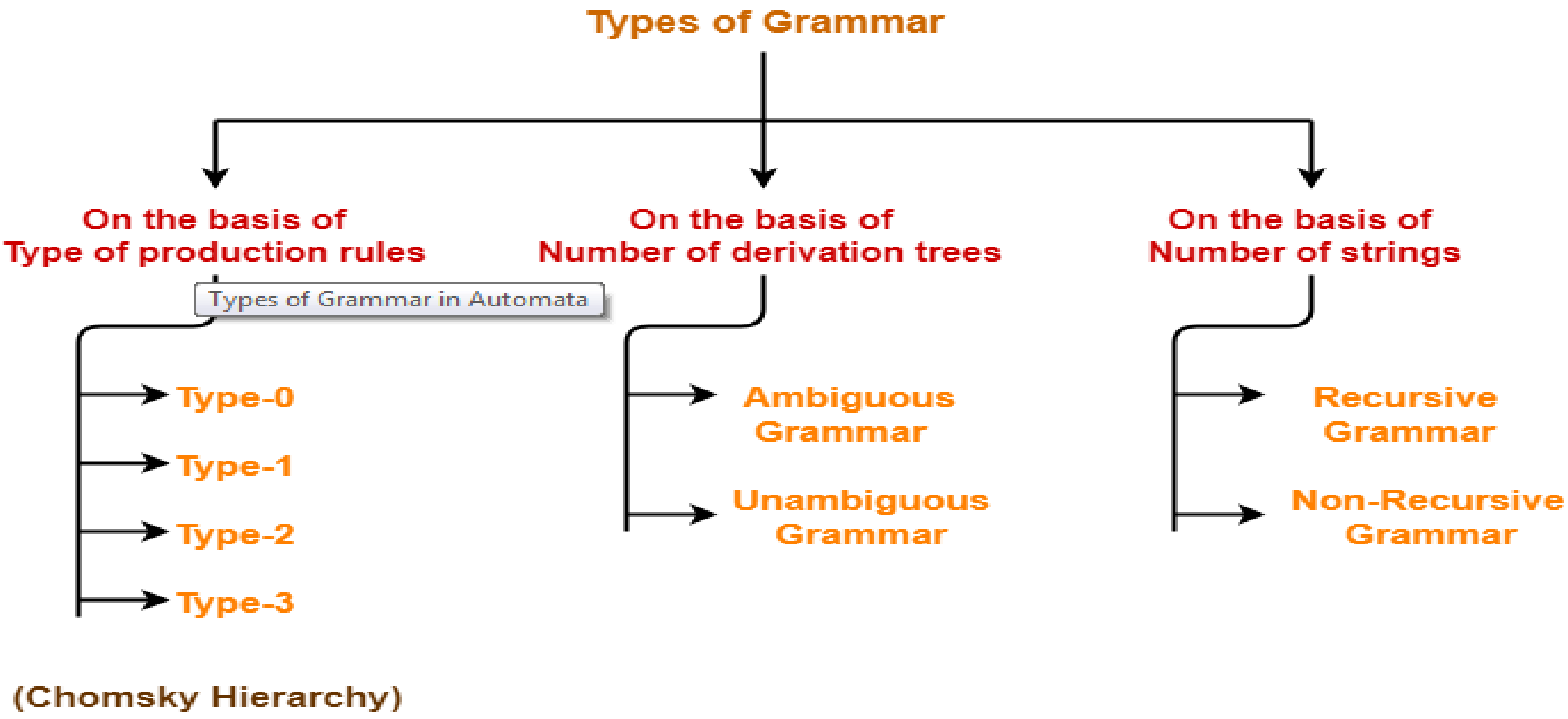
V = {S,A,B}

T = {a,b}

P = {S→ABA, A→BB, B→ab, AA→b}

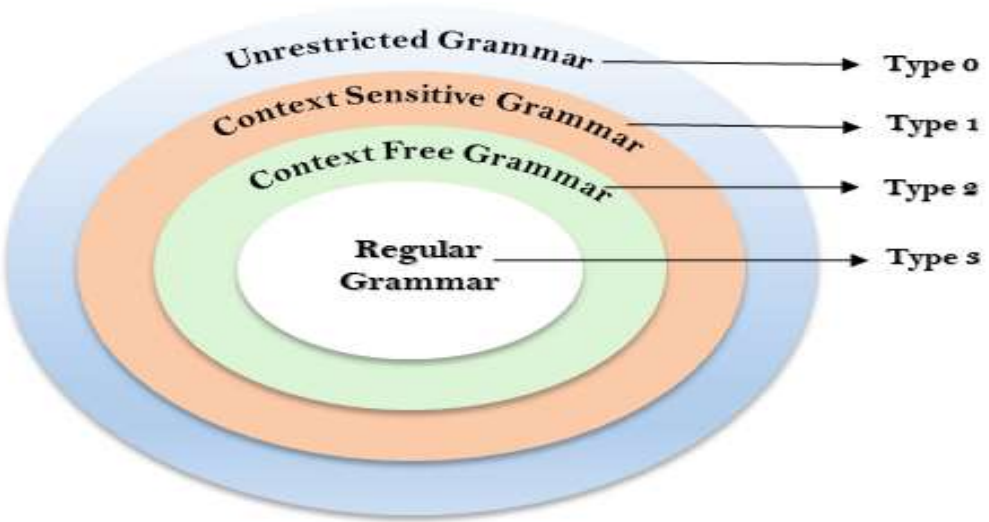
S = {S}

ypes of Grammar



Chomsky Hierarchy

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton



Type 3 ⊆ Type2, Type 1, Type 0
Type 2 ⊆ Type 1, Type 0
Type 1 ⊆ Type 0

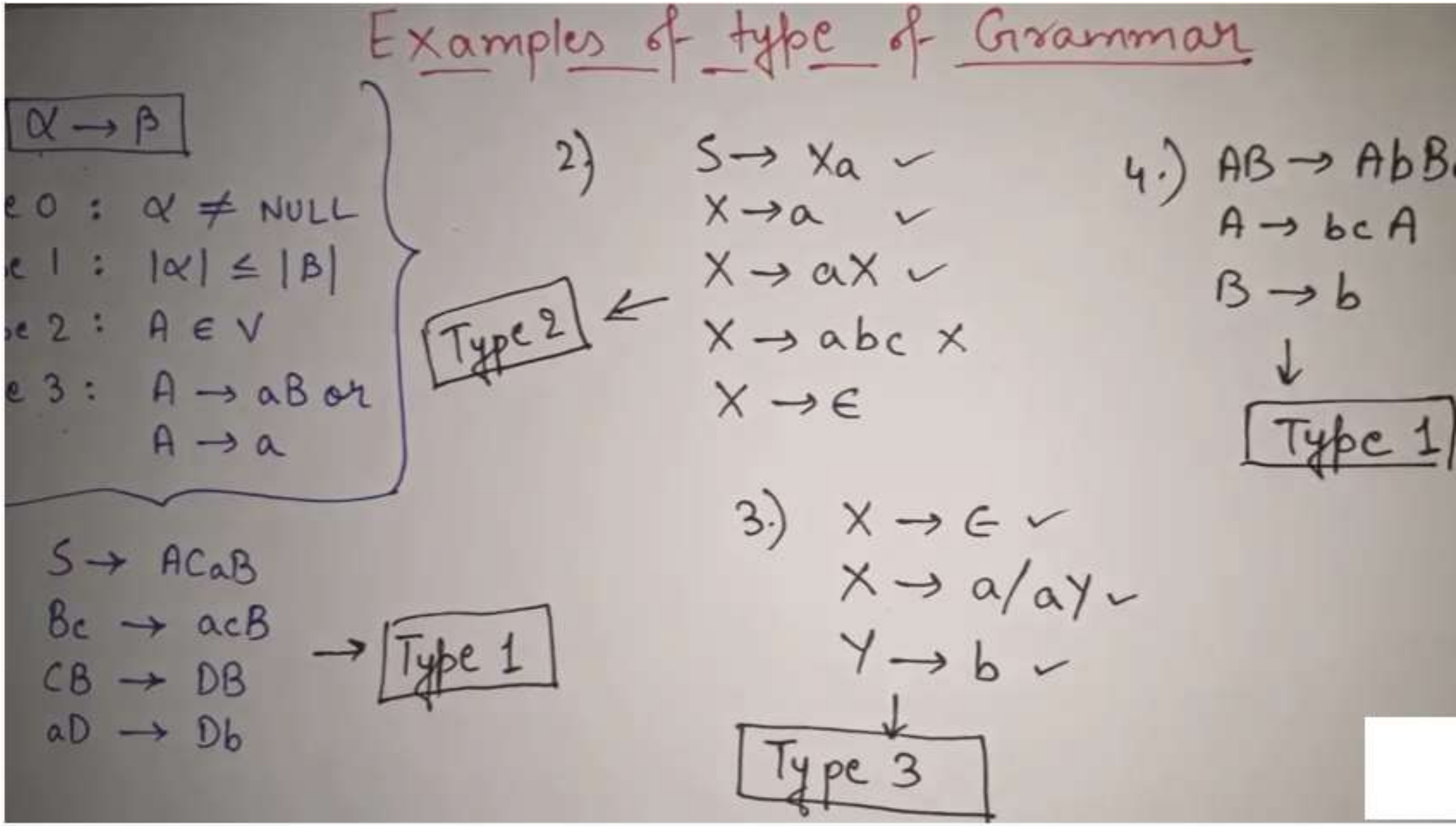
Chomsky Hierarchy

- Type 0 (Unrestricted)
  - α → β
  - α ∈ $(V+T)^+$ ← excluding ∈
  - β ∈ $(V+T)^*$ ← including ∈
  - α ≠ ∈
- Type 1 (Context Sensitive Grammar)
  - α → β
  - | α|<= | β|
- Type 2 (Context Free Grammar)
  - α → β
  - α ∈ V
  - β ∈ $(V+T)^*$
- Type 3 (Restricted)
  - V→VT*/T* (Right Regular Language) OR T*V/T* (Left Regular language)
  - Example: A→aB, A→ a

Chomsky Hierarchy

Rules

Type 0 → α ≠ ∈
Type 1 →| α|<= | β|
Type 2 → α ∈ V, β ∈ (V+T)*
Type 3 -> α->aB or α->a



References

- John E. Hopcroft and Rajeev Motwani and Jeffrey D. Ullman, "Introduction to Automata Theory, Languages and Computation", Second Edition, Pearson Education, New Delhi, (2007) (UNIT-I)
- Linz P. An introduction to formal languages and automata. Sixth edition, Jones and Bartlett Publishers; 2016.(UNIT-I)
- Ramaiah k. Dasaradh "Introduction to Automata and Compiler Design " First Edition ,Prentice Hall India Learning Private Limited(2011)(UNIT-I to V)


Thank
you