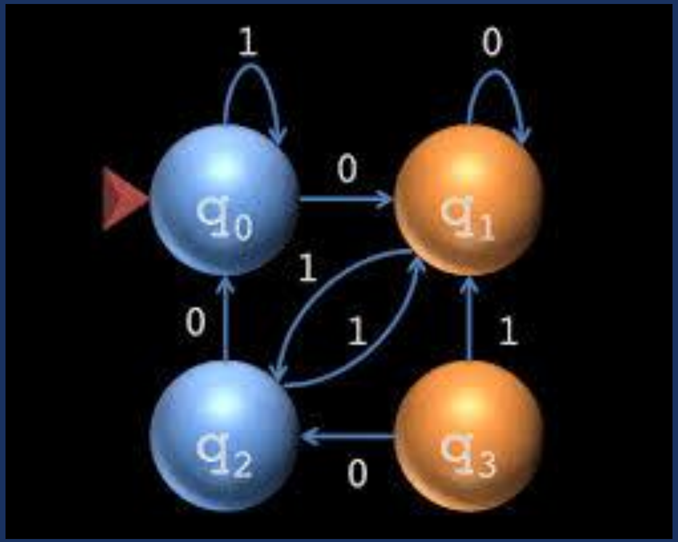
# CS3005 –THEORY OF AUTOMATA

## WEEK 1: INTRODUCTION TO FORMAL LANGUAGES & FINITE THEORY OF AUTOMATA





Syed Faisal Ali
sfaisal@nu.edu.pk
Computer Science 4E & 4F


SPRING 2025

# MAJOR PARTS IN THEORY OF AUTOMATA

Foundation

Pushdown Automata Theory

Turing Theory

# DETAILED COURSE OUTLINE

Foundation - 11

- Introduction to Theory of Automata.
- Languages.
- Recursive Definitions.
- Regular Expressions.
- Finite Automata.
- Transition Graphs.
- Kleene's Theorem.
- Finite Automata with Outputs.
- Regular Languages.
- Non-Regular Languages.
- Decidability

# DETAILED COURSE OUTLINE

**Pushdown Automata Theory - 7**

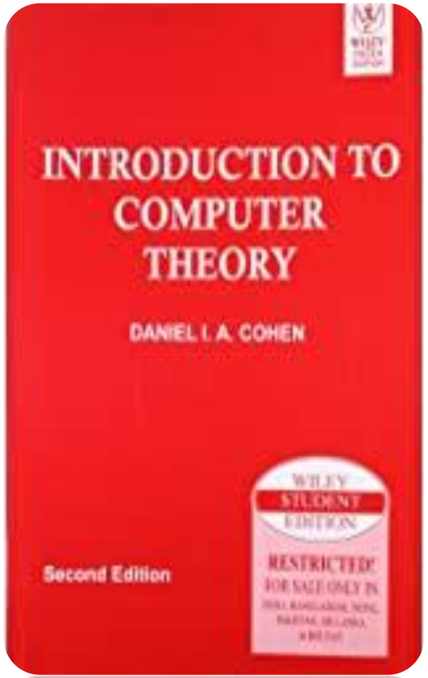
- Context Free Grammars.
- Grammatical Format.
- Pushdown Automata.
- CFG = PDA.
- Non-Context Free Languages.
- Context-Free Languages.
- Decidability

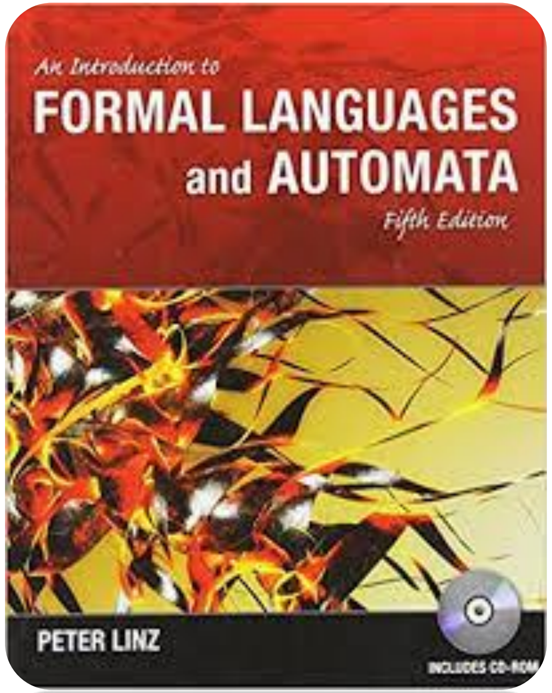
# DETAILED COURSE OUTLINE

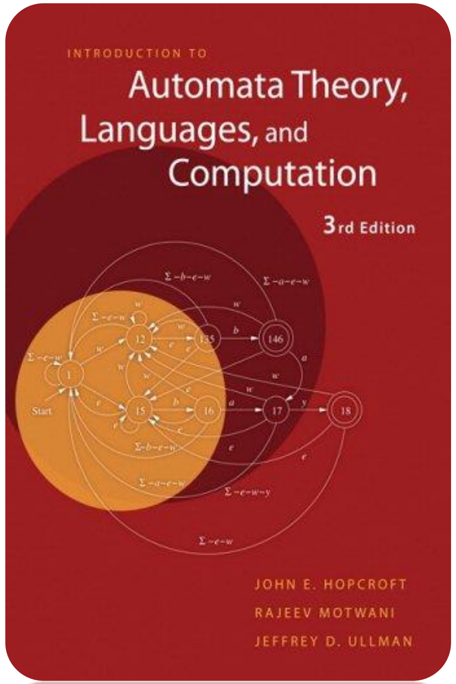
## Turing Theory - 7

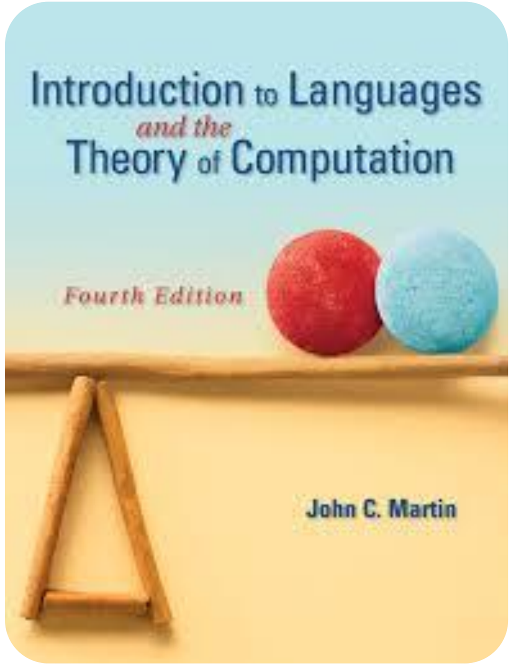
- Turing Machines.
- Post Machines.
- Minsky's Theorem.
- Variations on the TM.
- TM Languages.
- The Chomsky Hierarchy.
- Computers.

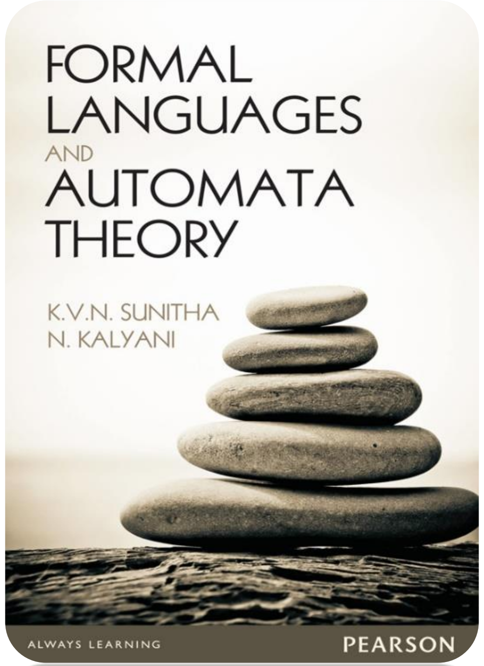
# BOOKS


INTRODUCTION TO COMPUTER THEORY
DANIEL I. A. COHEN
Second Edition
WILEY STUDENT EDITION
RESTRICTED!


An Introduction to
FORMAL LANGUAGES and AUTOMATA
Fifth Edition
PETER LINZ
INCLUDES CD-ROM


INTRODUCTION TO
Automata Theory, Languages, and Computation
3rd Edition
JOHN E. HOPCROFT
RAJEEV MOTWANI
JEFFREY D. ULLMAN


Introduction to Languages and the Theory of Computation
Fourth Edition
John C. Martin


FORMAL LANGUAGES AND AUTOMATA THEORY
K.V.N. SUNITHA
N. KALYANI
ALWAYS LEARNING
PEARSON

# MARKS DISTRIBUTION

- Mid term I & II – 30 Marks
- Final exam – 50 Marks
- Project – 10 Marks
- Quiz + Activities – 5 Marks (n-1)
- Assignments – 5 Marks (n-1)

# FORMAL LANGUAGE & FINITE THEORY OF AUTOMATA

- When we call our study the Theory of Formal Languages, the word "formal" refers to the fact that all the rules for the language are explicitly stated in terms of what strings of symbols can occur. No liberties are tolerated, and no reference to any "deeper understanding" is required. Language will be considered solely as symbols on paper and not as expressions of ideas in the minds of humans.
- In this basic mod The term "formal" used here emphasizes that it is the form of the string of symbols we are interested in, not the meaning. We begin with only one finite set of fundamental units out of which we build structures. We shall call this the alphabet.
- A certain specified set of strings of characters from the alphabet will be called the language. Those strings that are permissible in the language we call words, The symbols in language is not communication among intellects, but a game of symbols with formal rules.

# WHY STUDY FORMAL LANGUAGES & FINITE THEORY OF AUTOMATA?

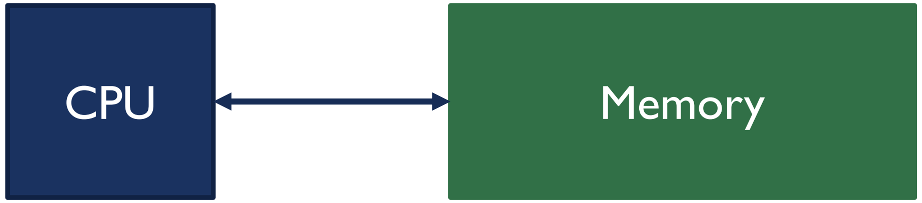
- As a computer professional you are dealing with logics. As we are dealing with high velocity, volume and multi type of file formatting we need high standards in computation machines.
- For the Processing of AI and Machine Learning, Deep Learning, Neural Networks we want to know how we are going to solve the problem and again our limitations come with hardware.
- Making a model in which we can able to understand that either the problem is solvable or unsolvable in given conditions or not.
- How we can compare that two different machines are performing same?
- How we are going to create new programming languages?
- Construction of new compilers to develop new programming languages.

# AUTOMATA

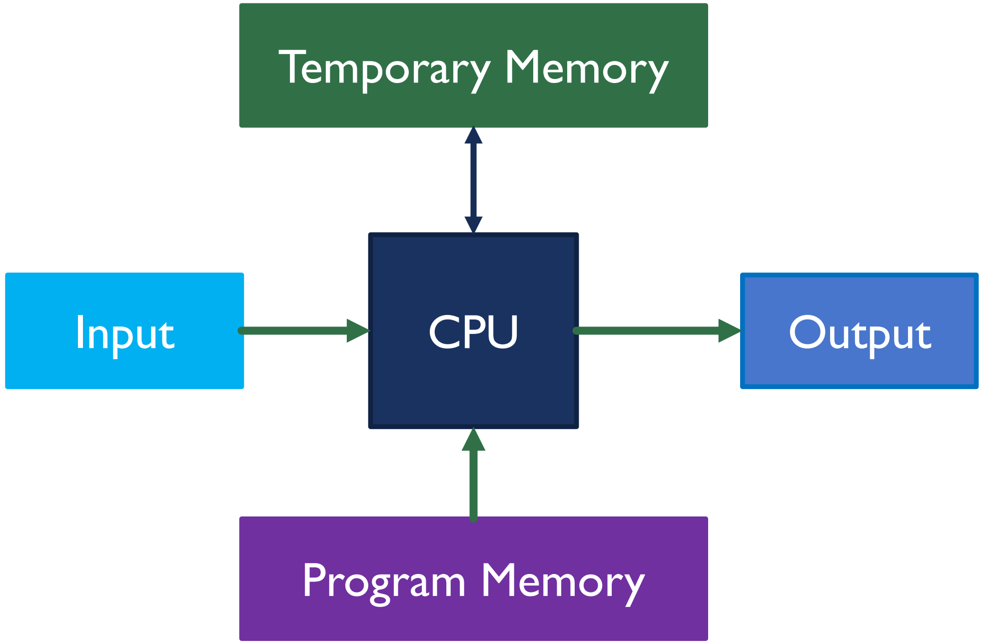
- The **word "automaton"** is the Latinization of **Greek** αὐτόματον, **automaton**, (neuter) "acting of one's own will".
- This **word** was first used by Homer to describe automatic door opening, or automatic movement of wheeled tripods.
- An automaton (Automata in plural) is an abstract self-propelled computing device which follows a predetermined sequence of operations automatically.
- An automaton with a finite number of states is called a **Finite Automaton (FA) or Finite State Machine (FSM).**

# COMPUTATION MODELS

A widely accepted model of computation

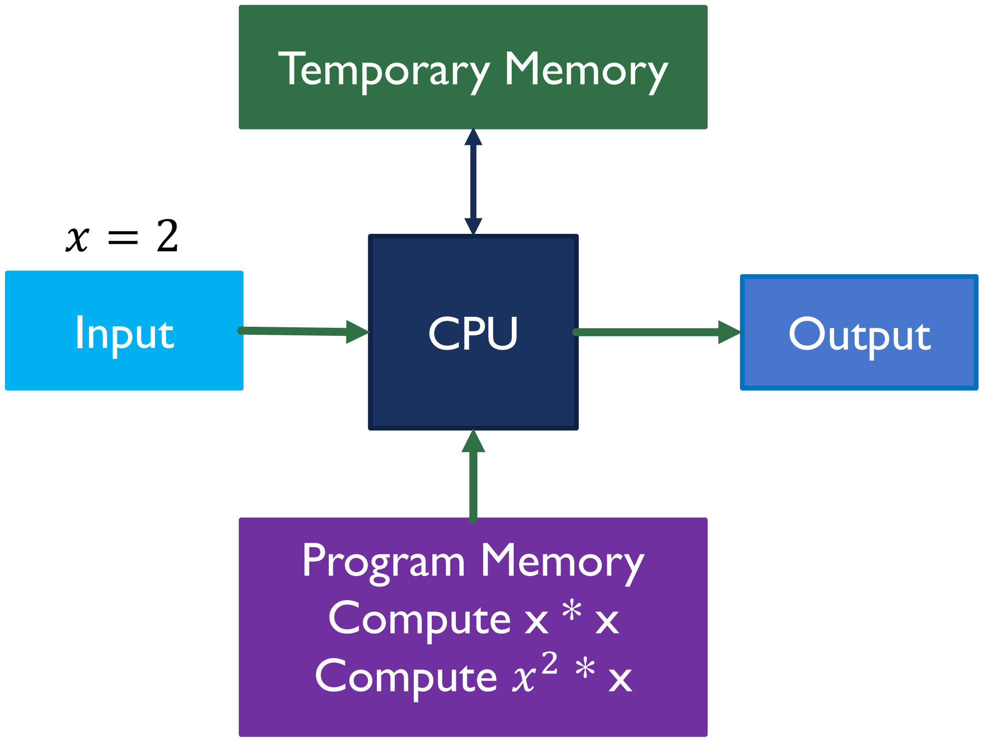


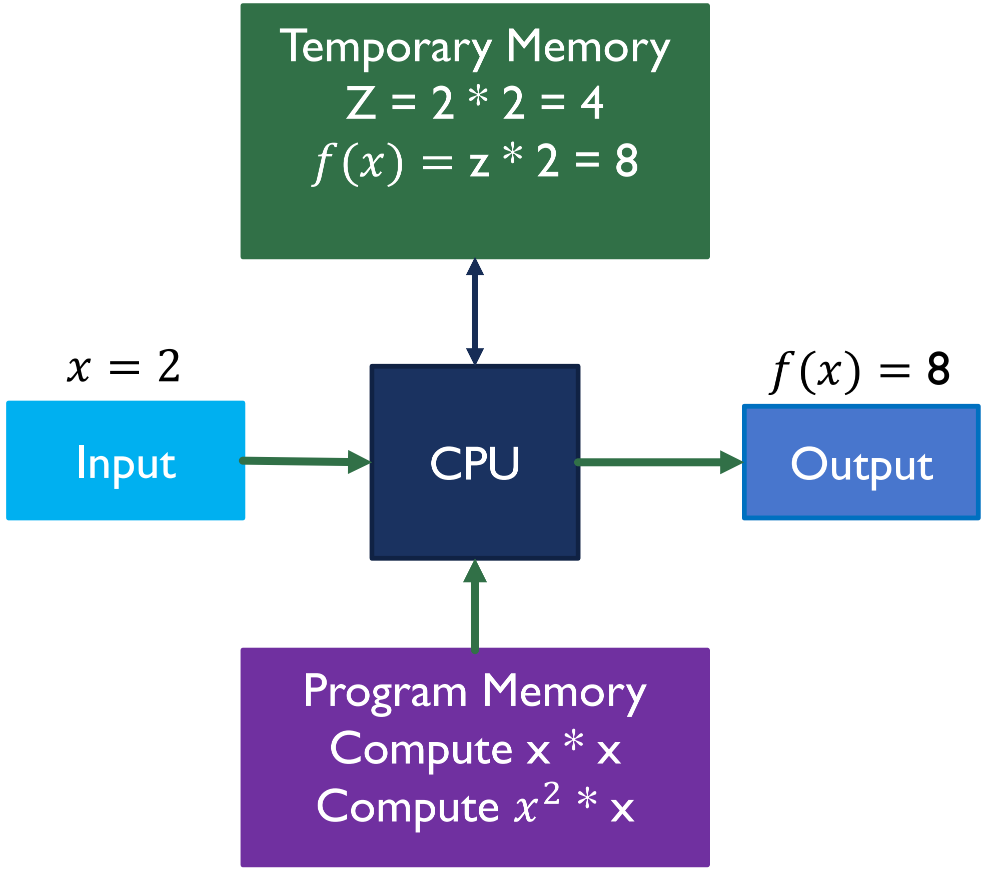

The different components of memory



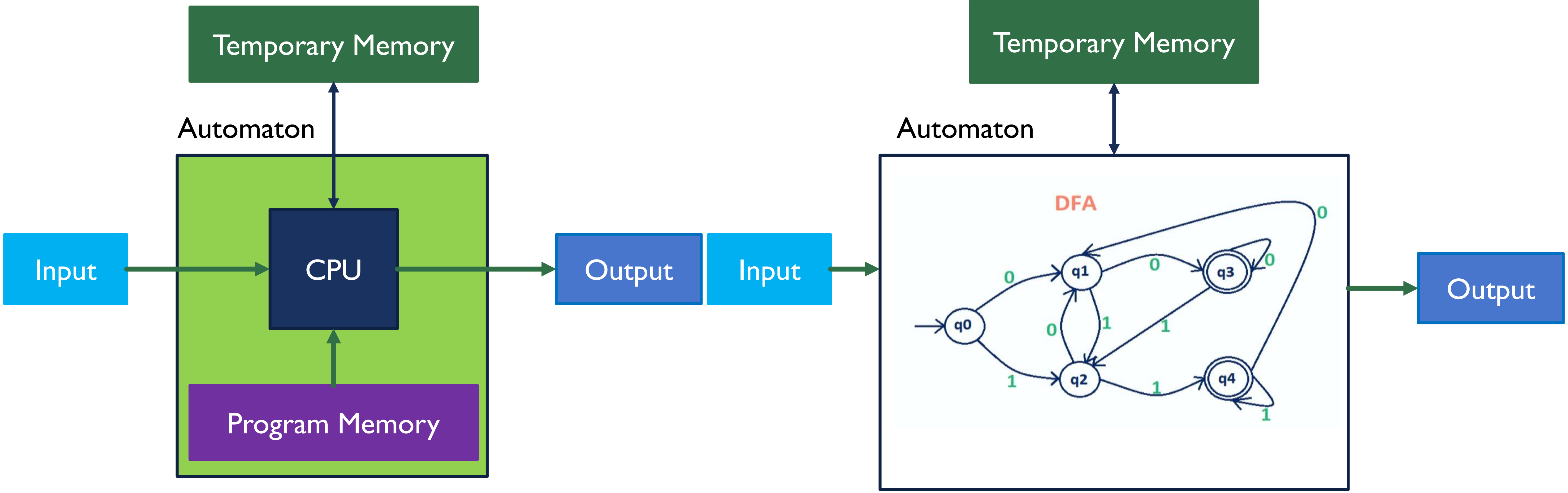
# PROCESSING THE MODEL

Example: $f(x) = x^3$



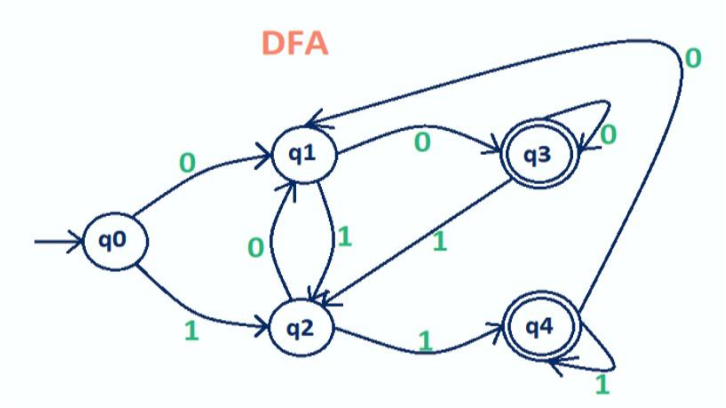
# AUTOMATON

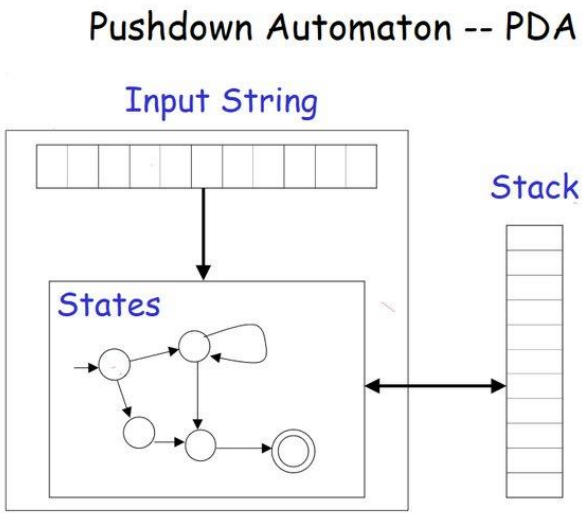


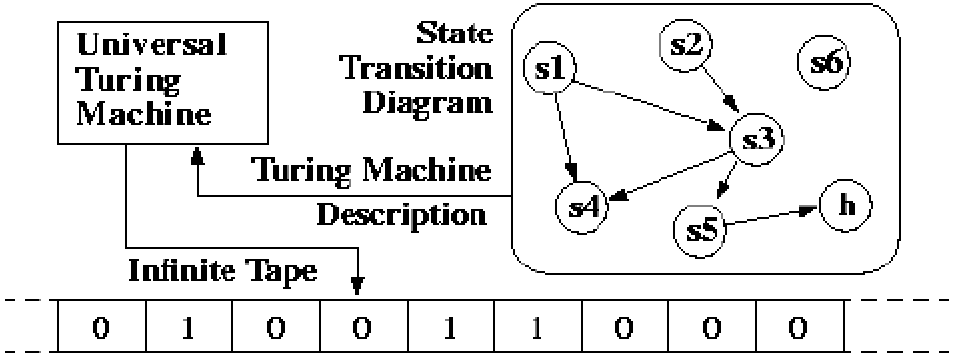

CPU + Program Memory = States + Transitions

# DIFFERENT KINDS OF AUTOMATA

| Type of Automata | Memory Type |
|---|---|
| Finite Automata | No Temporary Memory |
| Push Down Automata | Stack |
| Turing Machines | Random Access Memory |



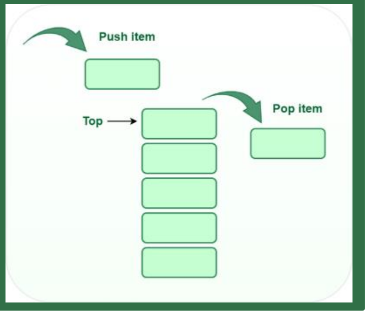



# COMPARISON OF PDA WITH TM

| Feature | Push Down Automata | Turing Machine |
|---|---|---|
| Tape | Has only one tape | Has infinite tape |
| Memory | Has a stack | Has infinite memory |
| Computational Power | Less powerful | More powerful |
| Acceptance of Languages | Accept Context Free Languages | Accepts all Languages |
| Type of Automata | Non-Deterministic | Deterministic |

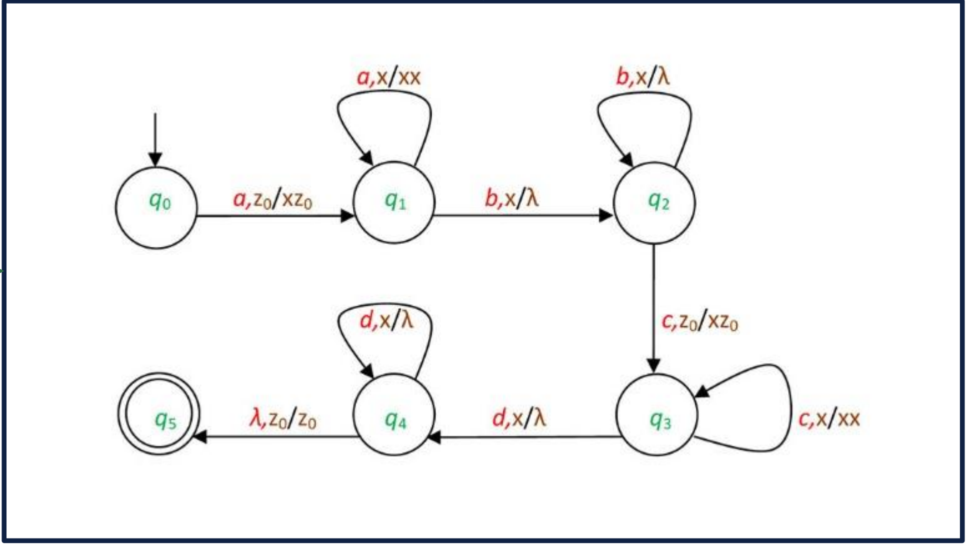
# PUSH DOWN AUTOMATA (PDA)

Temporary Memory




**Example:** Parsers for Programming Languages (medium computing power)

Push Down Automaton




Input

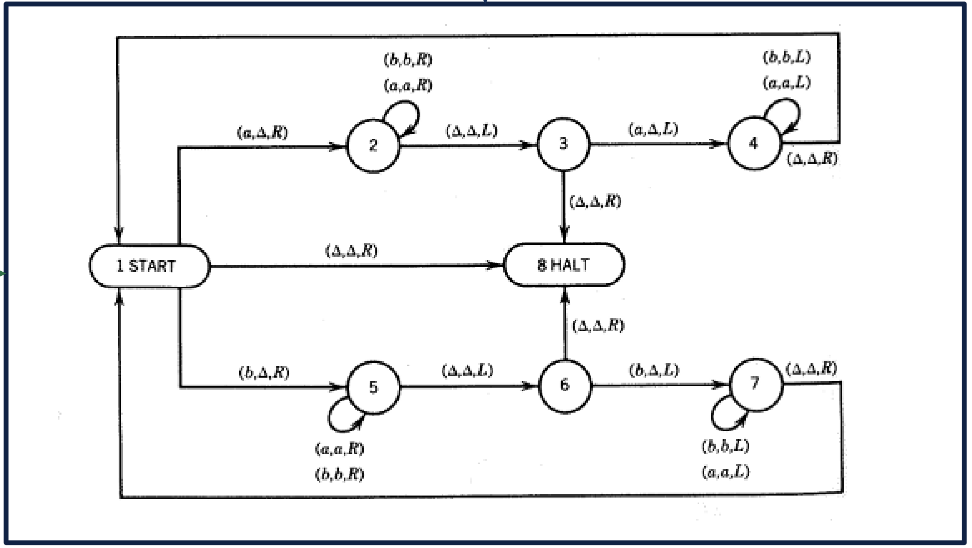
Output

# TURING MACHINE (TM)

Random Access Memory

**Example:** Any algorithm (highest known computing power)

Turing Machine




Input

Output

# POWER OF AUTOMATA

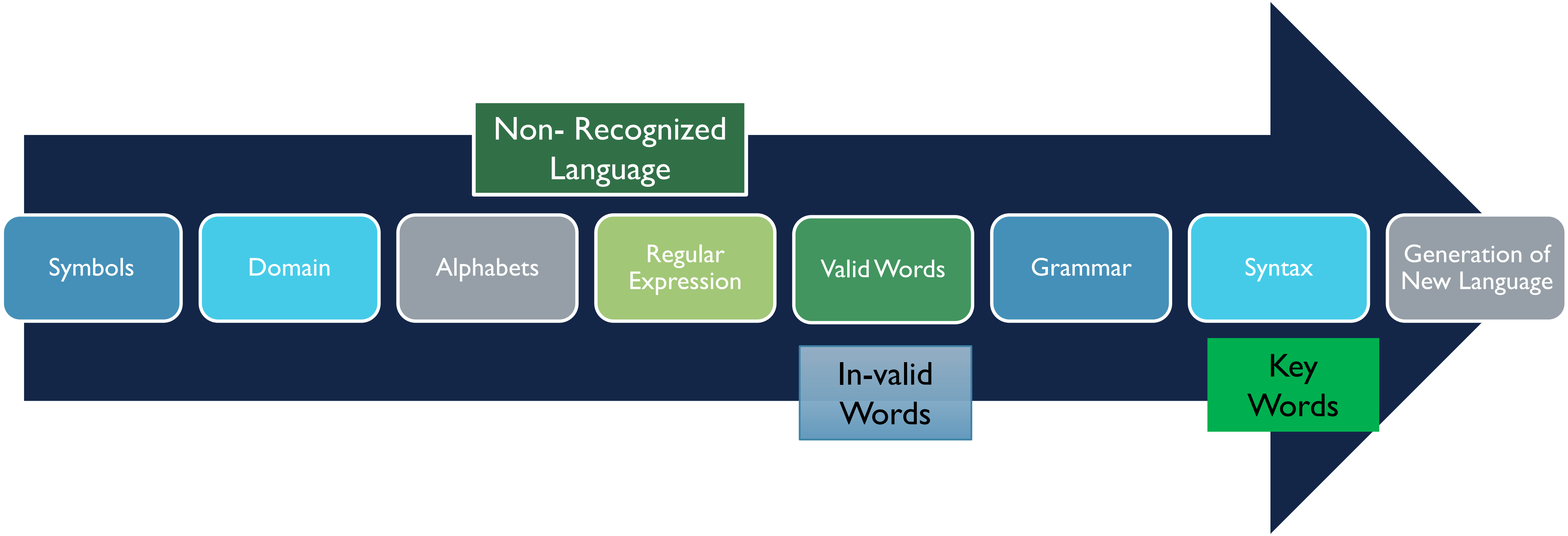
| Simple Problems | More Complex Problems | Hardest Problems |
|---|---|---|
| Finite Automata < | Push Down Automata < | Turing Machine |
| Less Power → | → | More Power |

Solve more computational problems

# THE MOST POWERFUL COMPUTATION MODEL

- Turing Machine is the most powerful known computation model
- Question: Can Turing Machine solve all computation problems?
- Answer: No (There are unsolvable problems too)

# WHAT WE DO IN AUTOMATA AS COMPUTER SCIENTIST?



# STRING & LANGUAGES

- The mathematical study of the "Theory of Computation" begins by understanding the Mathematics of strings of symbols.
- *Alphabet: It is defined as a finite set of symbols.*
- *Example : Roman alphabet {a, b, ...... z}.*
- "Binary Alphabet" {0, 1} is pertinent to the theory of computation.
- *String: A "string" over an alphabet is a finite sequence of symbols from that* alphabet, which is usually written next to one another and not separated by commas.
- (i) If $\sum_a = \{0, 1\}$ then 001001 is a string over $\sum_a$.
- (ii) If $\sum_b$= {a, b, c, ……z} *then axyrpqstcd is a string over* $\sum_b$.
- *Length of String: The "length" of a string is its length as a sequence. The* length of a string *w is written as |w|.*
- *Example: |10011| = 5*

# STRING & LANGUAGES

- ***Empty String**:The string of zero length is called the "empty string". This is* denoted by $\in_0$, or $\varepsilon_0\ or\ \lambda$ (Epsilon / Lambda).
- The empty string plays the role of 0 in a number system.
- ***Reverse String:** If w =* $w_1\ w_2\ w_3 \dots w_n$ *where each* $w_i \in \sum$, the reverse of w is $w_n\ w_{n-1}\ w_{n-2} \dots w_1$
- ***Substring:** z is a substring of w if z appears consecutively within w.*
- As an example, 'deck' is a substring of 'abcdeckabcjkl'.
- ***Concatenation:** Assume a string x of length m and string y of length n, the* concatenation of *x and y is written xy, which is the string obtained by* appending *y to the end of x, as in* $x_1\ x_2\ x_3 \dots x_m\ y_1\ y_2\ y_3 \dots y_n$ .

# SUFFIX , PREFIX AND LEXICOGRAPHICAL ORDER

- To concatenate a string with itself many times we use the "superscript" notation.

$$\overbrace{x\,x\;\ldots\;x}^{k} = x^{k}$$

- **Suffix:** *If w = xv for some x, then v is a suffix of w.*
- **Prefix:** *If w = vy for some y, then v is a prefix of w.*
- **Lexicographic ordering:** *The Lexicographic ordering of strings is the same as* the dictionary ordering, except that shorter strings precede longer strings.

# LANGUAGES

- A *language* is a subset of $\Sigma^*$ for some alphabet $\Sigma$.
- Example: The set of strings of 0's and 1's with no two consecutive 1's.
- L = {ϵ, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, . . . }
- A super set can be more complex and have all the members in the Universal set of 0 and 1.
- {0,1}* = {ϵ, 0, 1, 00, 01, 10, 11, 000, 001, . . . }
- Subtlety: 0 as a string, 0 as a symbol look the same.

# GENERATING LANGUAGE

- Let us consider some simple examples of languages. If we start with an alphabet having only one letter, the letter x,

$$\Sigma = \{x\}$$

- We can define a language by saying that any non empty string of alphabet characters is a word.

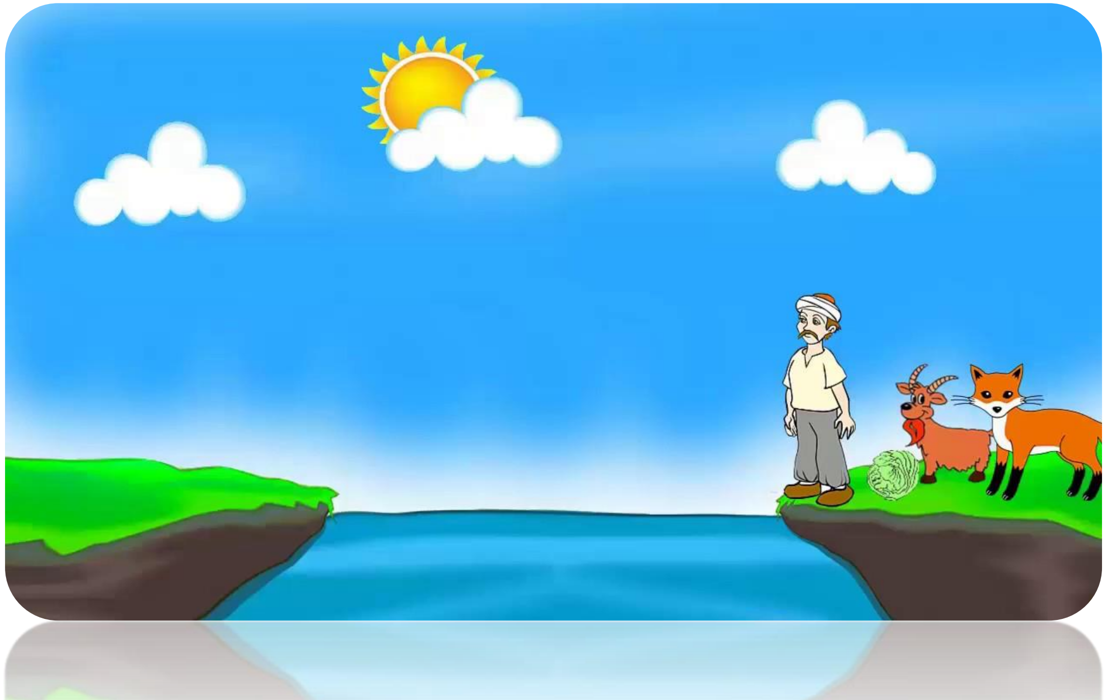
$$L1=\{x\ xx\ xxx\ xxxx...\}$$

- Or to write this in an alternate form:

$$L1 = \{x^n \text{ for } n = 1\ 2\ 3\ ...\}$$

# MAN, GOAT, WOLF AND CABBAGE

- Now consider a riddle in which we there is a man has a goat, a wolf, and  a cabbage. He has to go across the river by boat but he can only take one thing at a time. If he takes the cabbage but leaves the goat and the wolf together , the wolf will eat the goat. If he takes the wolf but leaves the goat and cabbage together, then the goat will eat the cabbage. How can he get the goat, the wolf and the cabbage across the river?



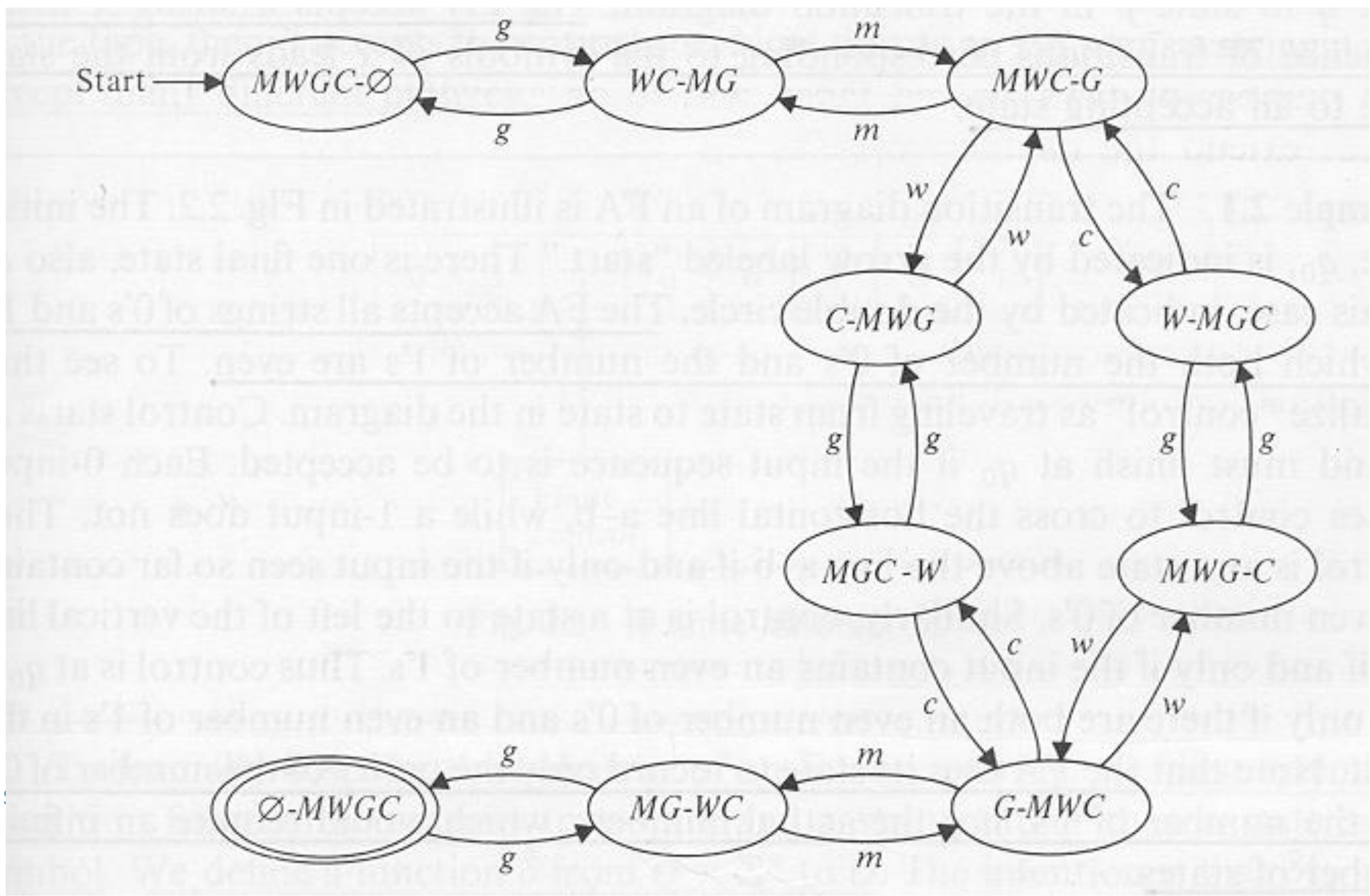
# SOLUTION

- The man first takes the goat across the river. He leaves the goat and returns to get the wolf. He leaves the wolf on the other side to but brings back the goat. He then takes the cabbage and leaves the goat. When he is on the other side he reaves the cabbage with wolf. The wolf will not eat the cabbage. The man goes back to pick up the goat and finally returns to the other side where the wolf and cabbage are waiting.
- Now the question is how you are going to solve this riddle using automata?

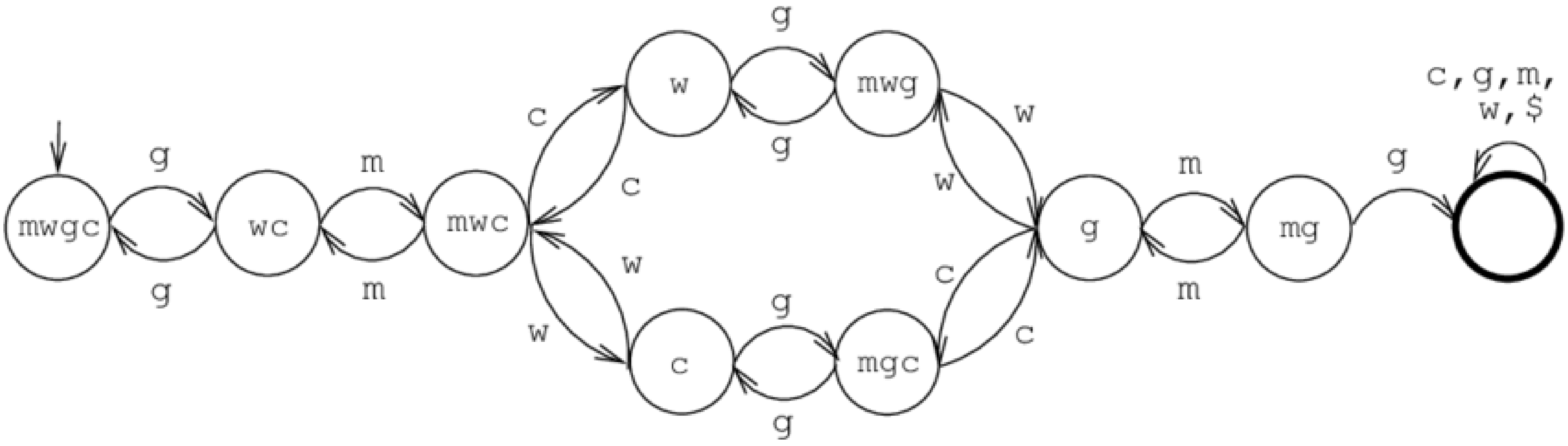
# SOLVE THIS RIDDLE USING AUTOMATA

- Step 1 : Find all the elements which are in the riddle.
- Step 2: Find the power set for the riddle.
- Step 3: Find all the constraints in your riddle.
- Step 4: Now identify your initial and final conditions.
- Step 6: Starting from initial condition try to find how we are going to connect to next step.
- Step 7: Continue step 6 until you reach final state.

# TRANSITION DIAGRAM FOR MAN, WOLF, GOAT AND CABBAGE



# GENERALIZED AUTOMATA FOR THE RIDDLE



# READING ASSIGNMENT

- Read what you have been taught today.
- Download book and read Chapter 1 & Chapter 2 from Daniel I A Cohen book.
- Update your python tool so we can start doing programming.
- Find one real problem and try to solve your problem by using automata.

# END OF LECTURE