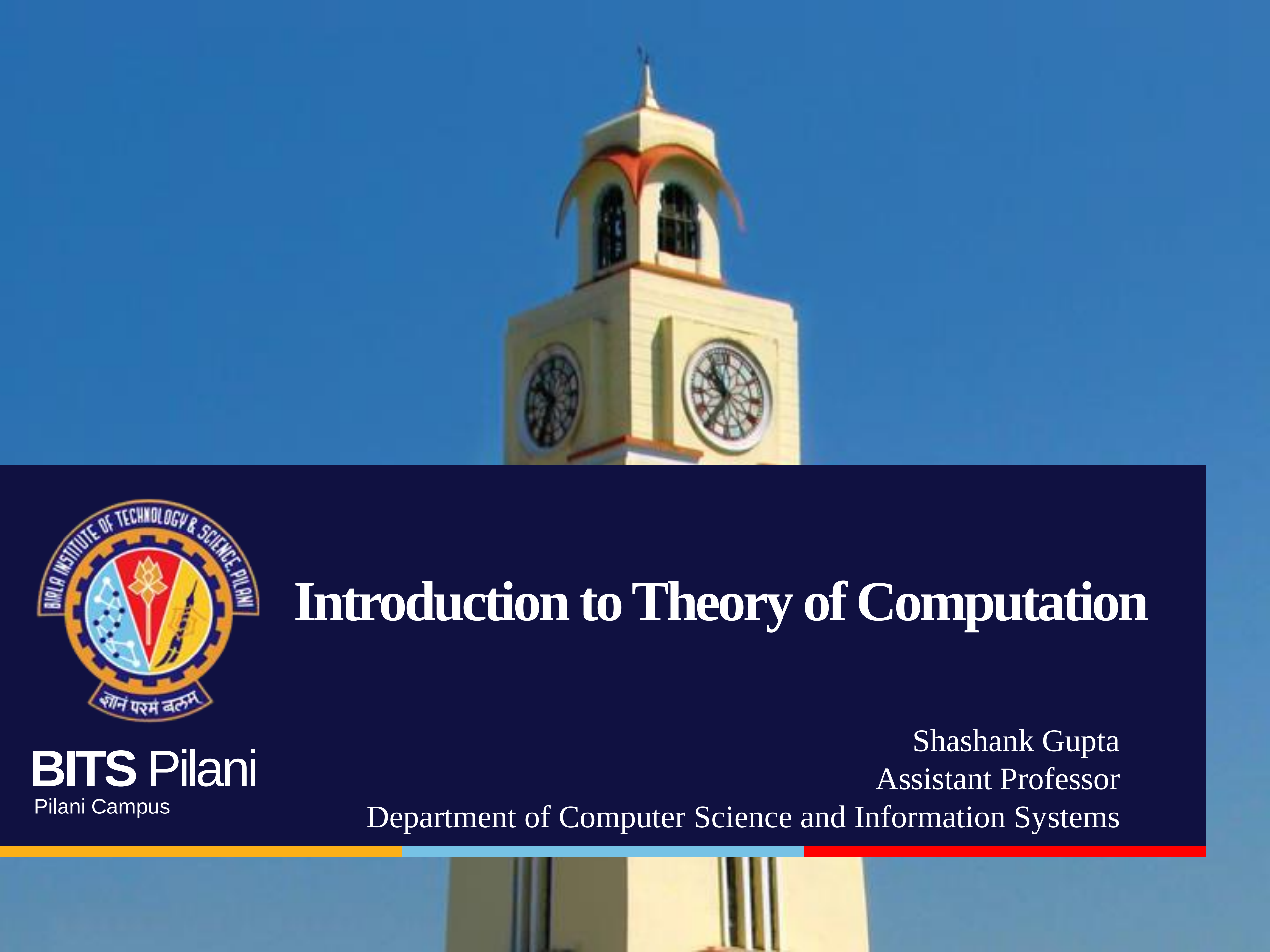


Introduction to Theory of Computation

BITS Pilani
Pilani Campus

Shashank Gupta
Assistant Professor
Department of Computer Science and Information Systems

Today’s Agenda

Key Points to be Covered

- Background and Motivation
- Informal Computational Models
- Finite Automata

References

Introduction to Theory of Computation, Michael Sipser

Introduction to Automata Theory, Languages and Computation, John Hopcroft, Rajeev Motwani and Jeffrey Ullman, Second Edition, Pearson,

- The course content (i.e. lecture slides, recorded lectures, pointers to reading material, etc.) will be available to you via the course website

Logistics of this Course

Please keep in touch with the course website https://nalanda.bits-pilani.ac.in/login/index.php by selecting the CS F351.

Feel free to discuss anything with us on the discussion forums/contact hours.

- Give feedback through out the semester.

Background and Motivation

This field of research was started by mathematicians and logicians in the 1930’s, when they were trying to understand the meaning of a “computation”

A central question asked was whether all mathematical problems can be solved in a systematic way.

- This course is about learning the fundamental capabilities and limitations of different computation models
- It is about mathematical properties of computer hardware and software.

Motivation Example

$$is\_prime(n) = \begin{Bmatrix} yes,\ if\ n\ is\ prime \\ no,\ if\ n\ is\ not\ prime \end{Bmatrix}$$

- **The definition of function does not point out in all cases to an algorithm to compute that function.**
- **This implies, one can possibly define/describe a function without having any idea to obtain the correct answer.**
- **In actual practice, there are many functions for which no algorithm exists.**

Goal

Basic Goal: To identify the class of functions which admit the algorithms to compute them.

- For function f there is an algorithm to compute f.

There are some other functions which do not admit any algorithm to compute them.

- In other words, there are some inputs for which no output can be generated from such functions.

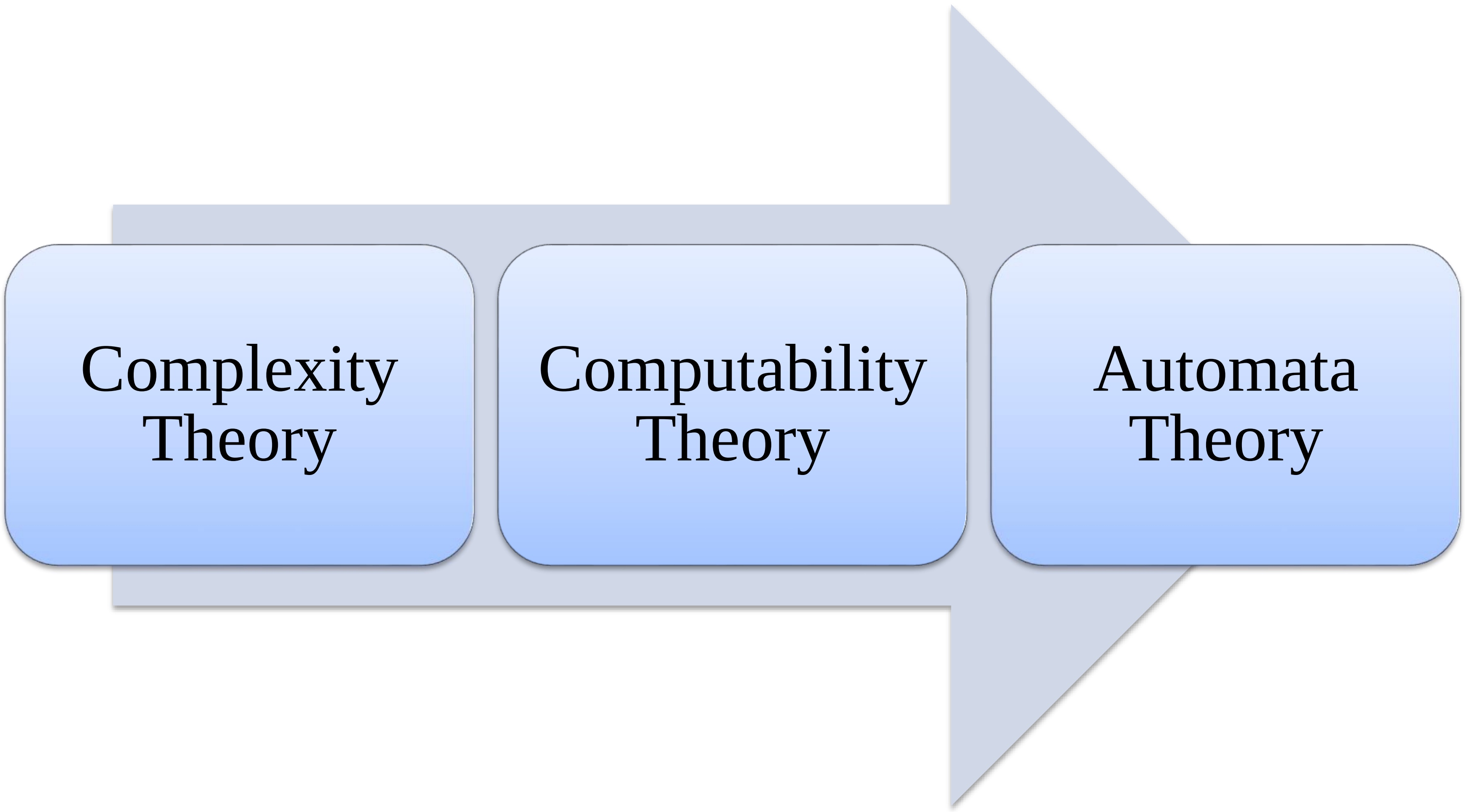
Learning Outcomes of this Course

This theory is very much relevant to practice, for example, in the design of new programming languages, compilers, string searching, pattern matching, computer security, artificial intelligence, etc.

This course helps you to learn problem solving skills. Theory teaches you how to think, prove, argue, solve problems, express, and abstract.

- **This theory simplifies the complex computers to an abstract and simple mathematical model, and helps you to understand them better.**

Areas of Theory of Computation



Complexity Theory

The main question asked in this area is

- What makes some problems computationally hard and other problems easy ?

Classify problems according to their degree of “difficulty”.

- Give a rigorous proof that problems that seem to be “hard” are really “hard”.

Computability Theory

Some of the fundamental mathematical problems cannot be solved by a “computer”

Classify problems as being solvable or unsolvable.

Complexity Theory vs Computability Theory

Complexity Theory

- In complexity theory, the objective is to classify problems as easy ones and hard ones.

Computability Theory

- The classification of problems is by those that are solvable and those that are not.

Automata

An **automaton** (**Automata** in plural) is an abstract self-propelled computing device which follows a predetermined sequence of operations automatically.

Examples of such models are Finite Automata, Pushdown Automata, Linear Bounded Automata and Turing Machine

Automata Theory

Deals with definitions and properties of different types of "**Computation Models**".

Do these models have the same power, or can one model solve more problems than the other?

Automata Theory

Finite Automata (FA)

A computational model with finite number of states.

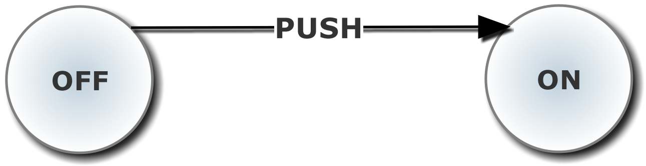
A finite representation of a formal language that may be an infinite set

Informal Examples of FA

- Electric Switch

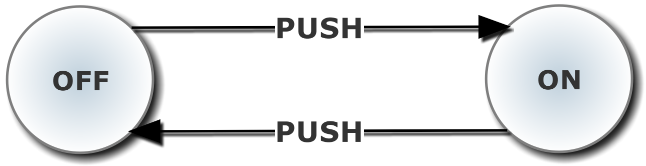
Informal Examples of FA

- Electric Switch



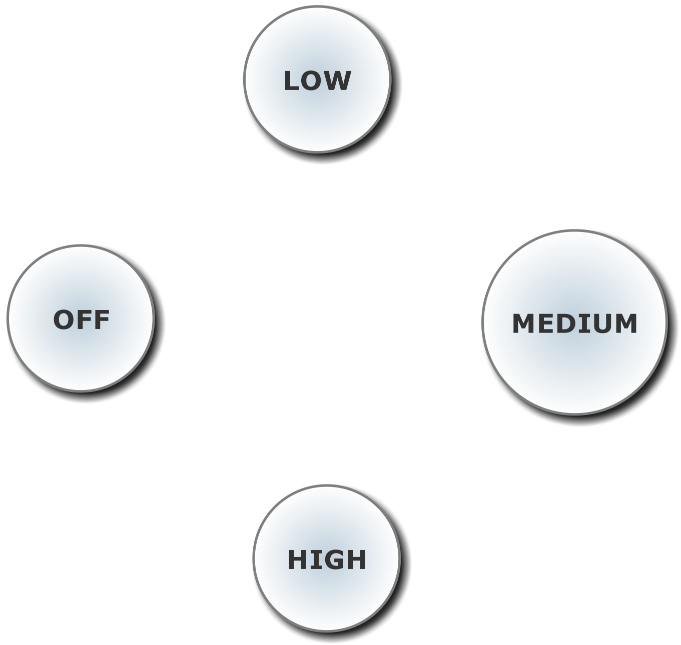
Informal Examples of FA

- Electric Switch



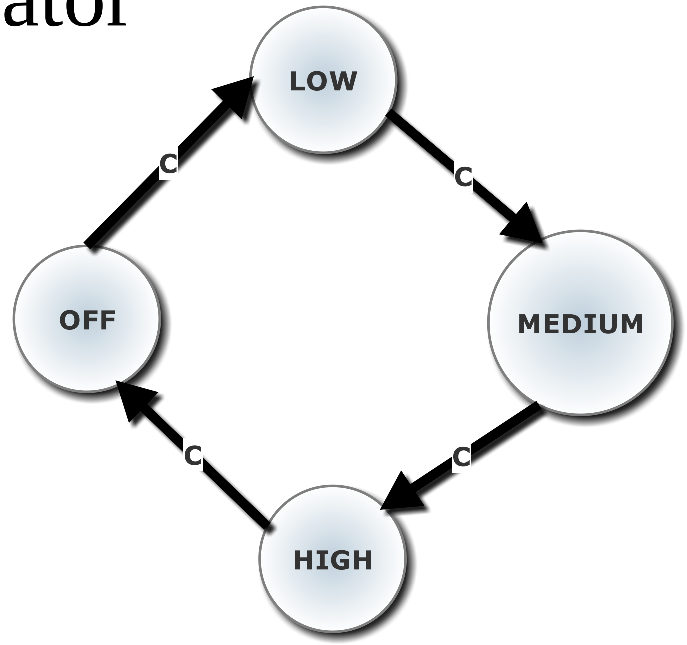
More Informal Examples

- Fan Regulator



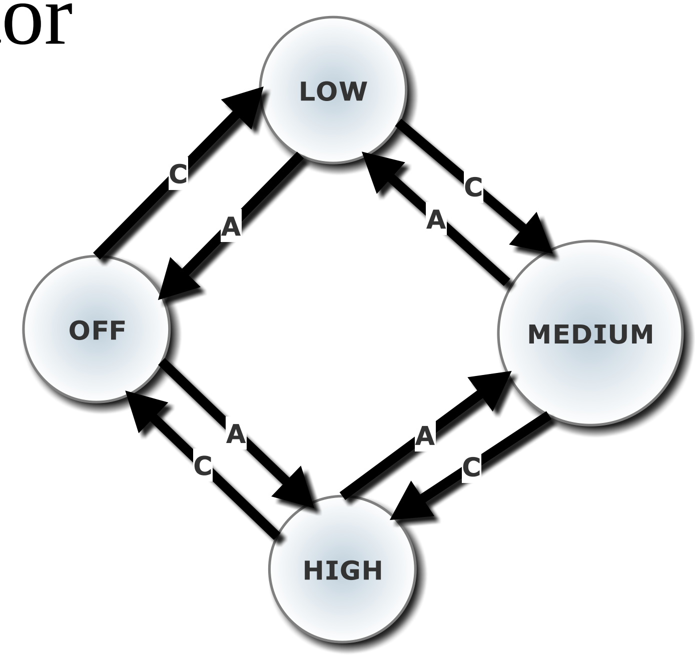
More Informal Examples

- Fan Regulator


$$\begin{tikzcd}
 & LOW \arrow[dr, "C"] & \\
OFF \arrow[ur, "C"] & & MEDIUM \arrow[dl, "C"] \\
 & HIGH \arrow[ul, "C"] &
\end{tikzcd}$$

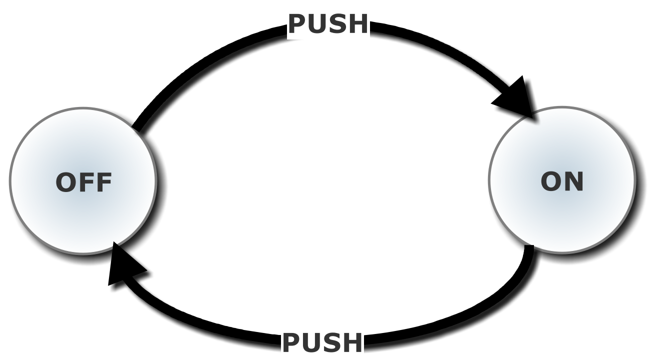
More Informal Examples

- Fan Regulator

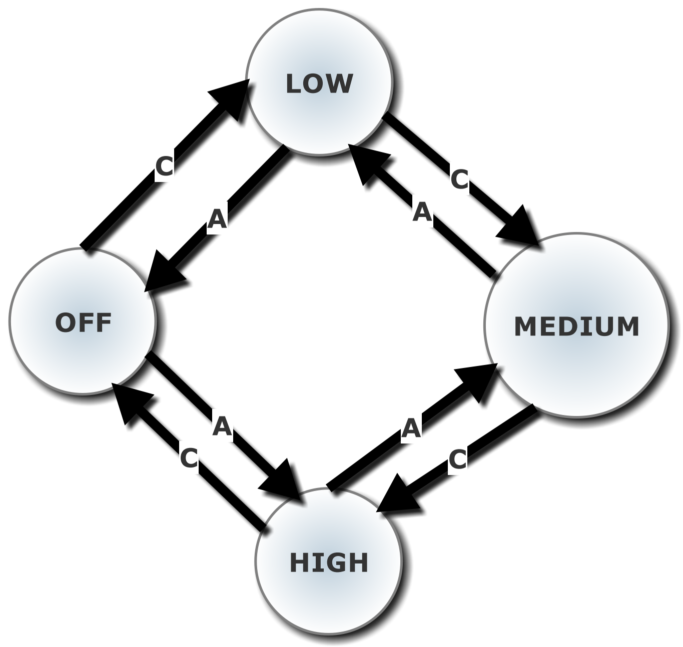


Informal Examples of FA

ELECTRIC SWITCH



FAN REGULATOR



Basic Terminologies

Symbol: Symbol is the smallest building block, which can be any alphabet, letter or any picture.

For example: 0, 1, 2, a, b, ------

Basic Terminologies (Continued….)

Input Alphabet (Σ): Any finite non-empty set of symbols.

For example: $\Sigma = \{0, 1\}, \{a, b\}$

Basic Terminologies (Continued….)

String is a ***finite*** sequence of symbols from some input alphabet **Σ**.

- String is generally denoted as ***w*** and length of a string is denoted as |***w***|.
- For e.g. Number of Strings (of length 2) that can be generated over the alphabet **Σ** = {0,1} is 00, 01, 10, 11
- Hence, Length of String |w| = 2, Number of Strings = 4

Empty string is the string with zero occurrence of symbols, represented as ∈ (Epsilon)

Input Alphabet (Σ)

$\Sigma^i = \{w \mid w \text{ is a string over } \Sigma \text{ and } |w| = i$

- Consider Σ={a,b}, then $\Sigma^2 = \{aa, ab, ba, bb\}$

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

- Note that, Σ* will contain ∈ (epsilon) while, Σ^+ does not include ∈

Basic Terminologies (Continued….)

Language (L): A Language L over an input alphabet (Σ) is a subset of Σ^*

For e.g. if $\Sigma = \{0, 1\}$ then $L = \{\in, 0, 1, 00, 101, ----\}$

- One special language is Σ^* , which is the set of all possible strings generated over the alphabet Σ . For example, if $\Sigma = \{a, b, c\}$ then $\Sigma^* = \{\in, a, b, c, aa, ab, ac, ba, \ldots, aaaaabbbaababa, \ldots\}$

Some More Basic Terminologies

Formal Language: A formal language consists of strings whose letters are taken from an alphabet and are well-formed according to a specific set of rules.

Automaton: It is a machine which is used to recognize the formal languages.

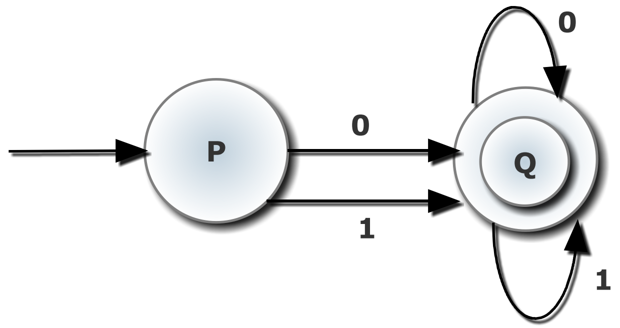
- Automaton are often classified by the different classes of formal languages that they are able to recognize. Now these different classes of formal languages and their corresponding automaton will be discussed in the future discussions.

Finite Automaton (FA)

FA	
	A mathematical model that consists of finite set of states and related transitions.
	States are generally represented by circles (labelled with state name).
	Transitions are usually represented by directional arrows labelled with input alphabet symbols.

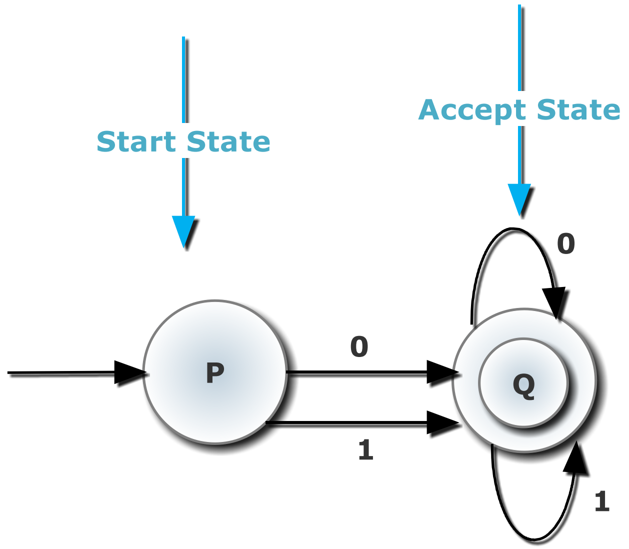
Representation of FA

- A Finite Automaton over Σ = {0,1}



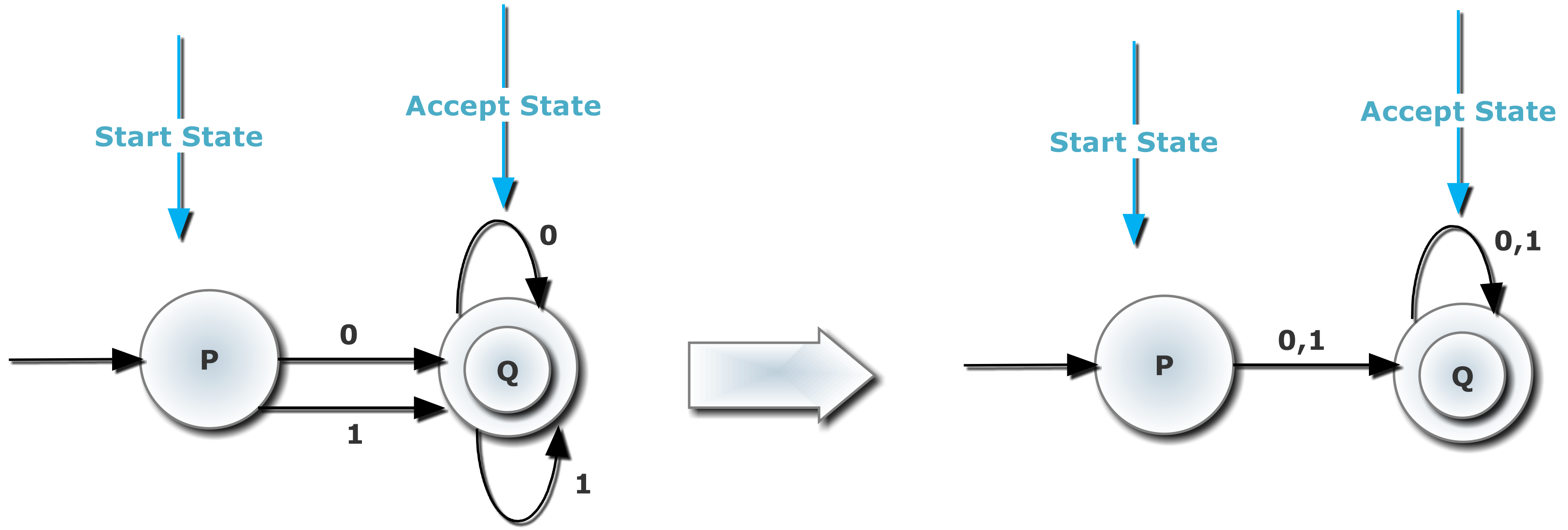
Representation of FA (Continued…)

- A Finite Automaton over Σ = {0,1}



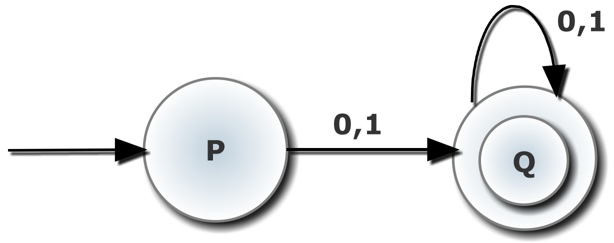
Representation of FA (Continued…)

- A Finite Automaton over Σ = {0,1}



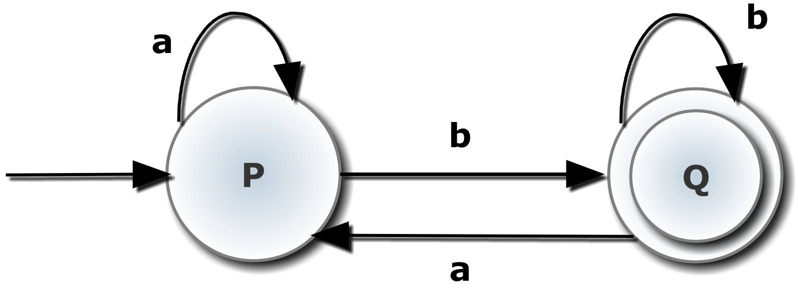
Language Acceptance Mechanism by FA

- FA accepts only those strings in the language L which ends up in any of its accept states.



L ={0, 1, 00, 11, 010, 110, 11110001100, ---------}

Language Acceptance Mechanism by FA (Continued…..)



Here, Language accepted by the above FA is L = {b, ab, aab, baabb,------}

However, strings like a, aa, ba ------- are not going to be accepted by this FA.