Introduction to ToC

BY:

Dr. Neethi M V

Assistant Professor

Dept. of CSE-DS

ATMECE, Mysuru

Module 1 – Introduction

Introduction to Finite Automata, Structural Representations, Automata and Complexity. The Central
Concepts of Automata Theory. Deterministic Finite Automata, Nondeterministic Finite Automata, An Application: Text Search, Finite Automata with Epsilon-Transitions.
TEXT BOOK: Sections 1.1, 1.5, 2.2,2.3,2.4,2.5

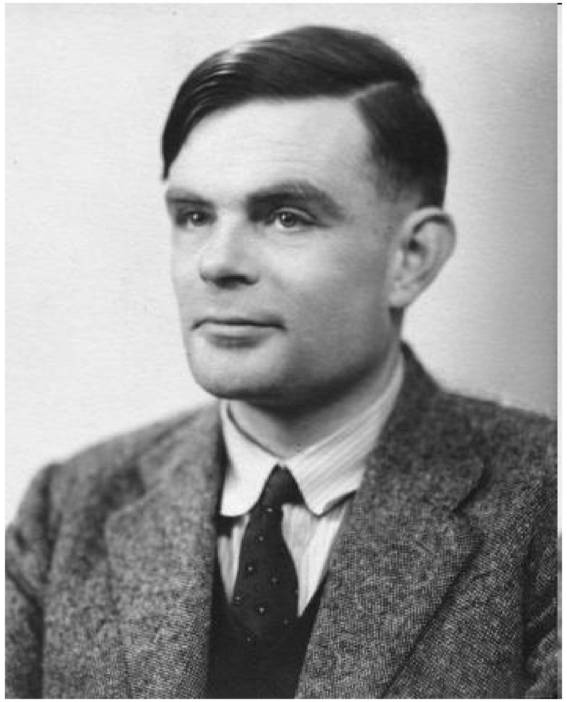
What is Automata Theory?

- *Study of abstract computing devices, or "machines"*
- Automaton = an abstract computing device
  - Note: A "device" need not even be a physical hardware!
- A fundamental question in computer science:
  - Find out what different models of machines can do and cannot do
  - The *theory of computation*
- Computability vs. Complexity

(A pioneer of automata theory)

Alan Turing (1912-1954)

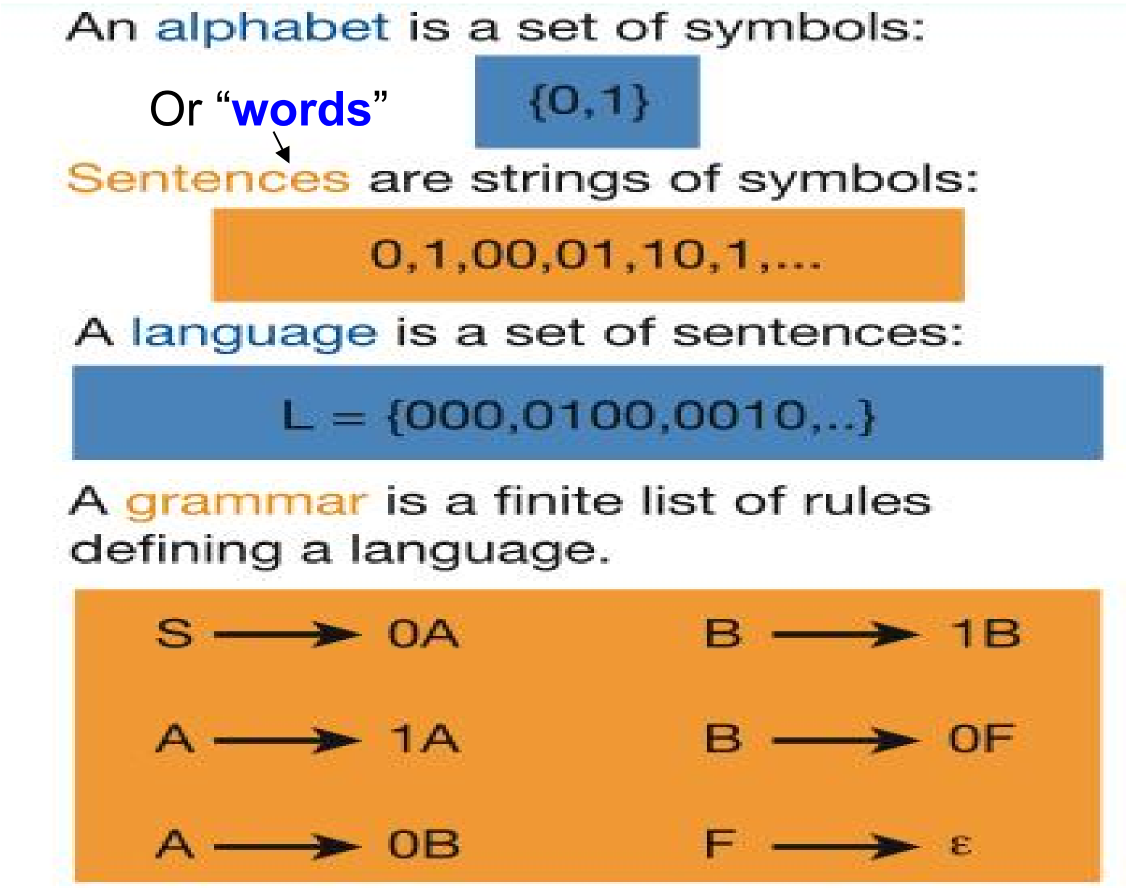
- Father of Modern Computer Science
- English mathematician
- Studied abstract machines called ***Turing machines*** even before computers existed
- Heard of the Turing test?



Theory of Computation: A Historical Perspective

1930s	• Alan Turing studies Turing machines • Decidability • Halting problem
1940-1950s	• "Finite automata" machines studied • Noam Chomsky proposes the "Chomsky Hierarchy" for formal languages
1969	Cook introduces "intractable" problems or "NP-Hard" problems
1970-	Modern computer science: compilers, computational & complexity theory evolve

Languages & Grammars



An alphabet is a set of symbols:

Or “**words**”

{0,1}

Sentences are strings of symbols:

0,1,00,01,10,1,...

A language is a set of sentences:

L = {000,0100,0010,..}

A grammar is a finite list of rules defining a language.

$S \longrightarrow 0A$ $\quad$ $B \longrightarrow 1B$

$A \longrightarrow 1A$ $\quad$ $B \longrightarrow 0F$

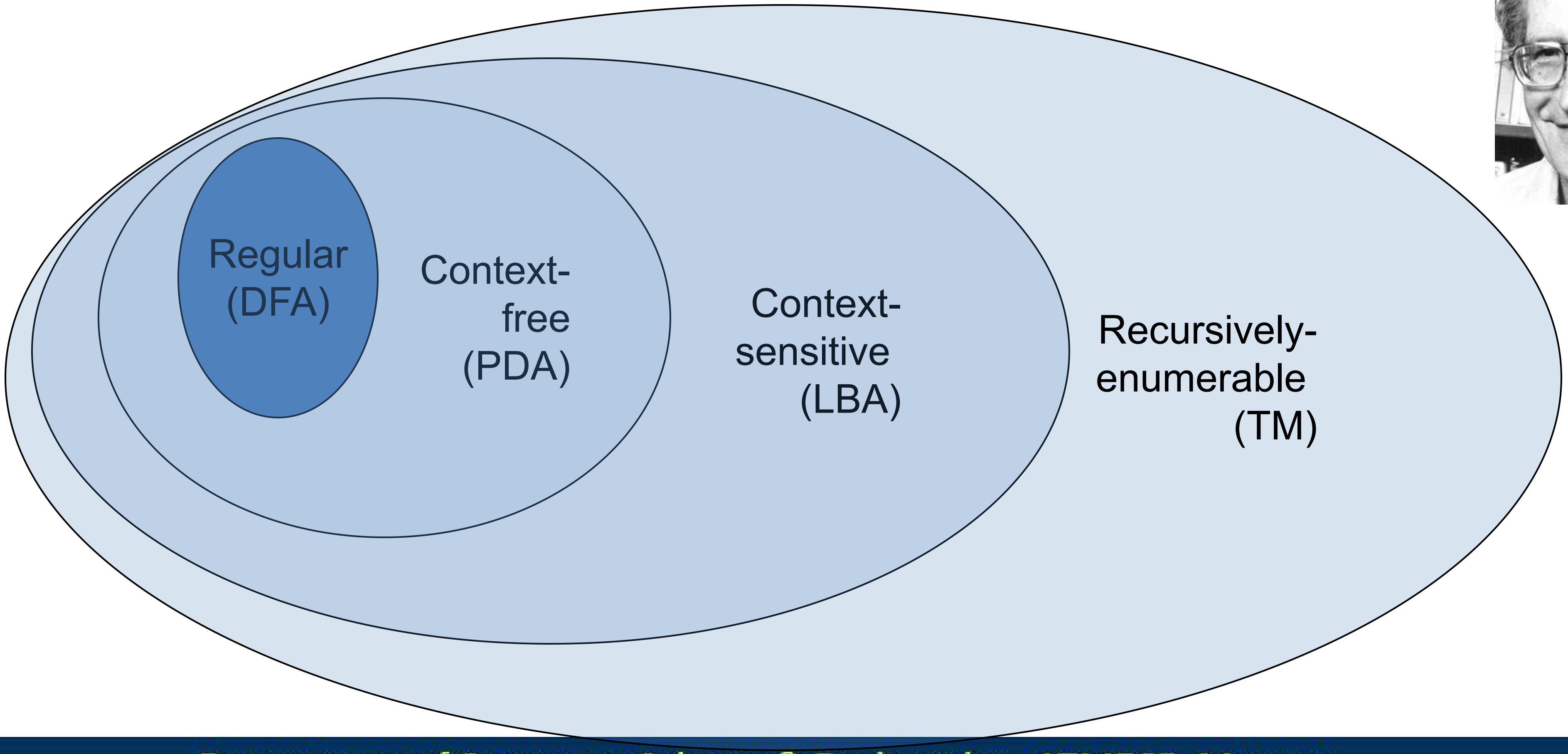
$A \longrightarrow 0B$ $\quad$ $F \longrightarrow \varepsilon$

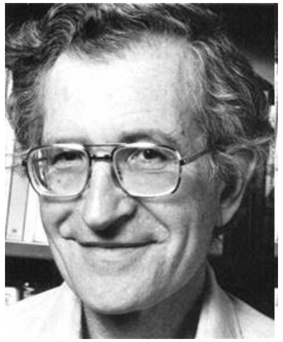
Image source: Nowak et al. Nature, vol 417, 2002

- Languages: *“A language is a collection of sentences of finite length all constructed from a finite alphabet of symbols”*
- Grammars: *“A grammar can be regarded as a device that enumerates the sentences of a language”* - nothing more, nothing less
- *N. Chomsky, Information and Control, Vol 2, 1959*

The Chomsky Hierachy

- A containment hierarchy of classes of formal languages





The Central Concepts of Automata Theory

Alphabet

An alphabet is a finite, non-empty set of symbols

- We use the symbol ∑ (sigma) to denote an alphabet
- Examples:
  - Binary: ∑ = {0,1}
  - All lower case letters: ∑ = {a,b,c,..z}
  - Alphanumeric: ∑ = {a-z, A-Z, 0-9}
  - DNA molecule letters: ∑ = {a,c,g,t}
  - …

Strings

A string or word is a finite sequence of symbols chosen from ∑

- ***Empty string is ε (or "epsilon")***

- Length of a string *w,* denoted by "|*w*|", is equal to the *number of (non- ε) characters in the string*
  - *E.g., x = 010100* *|x| = 6*
  - *x = 01 ε 0 ε 1 ε 00 ε* *|x| = ?*

– *xy* = concatentation of two strings *x* and *y*

Powers of an alphabet

Let ∑ be an alphabet.

- $\sum^{k}$ = the set of all strings of length *k*
- $\sum^{*} = \sum^{0} \cup \sum^{1} \cup \sum^{2} \cup \ldots$
- $\sum^{+} = \sum^{1} \cup \sum^{2} \cup \sum^{3} \cup \ldots$

Languages

L is a said to be a language over alphabet ∑, only if $L \subseteq \Sigma^*$

➔ this is because Σ^* is the set of all strings (of all possible length including 0) over the given alphabet ∑

Examples:

1. Let L be *the* language of all strings consisting of *n* 0's followed by *n* 1's:
   L = {ε, 01, 0011, 000111,…}
2. Let L be *the* language of all strings of with equal number of 0's and 1's:
   L = {ε, 01, 10, 0011, 1100, 0101, 1010, 1001,…}

⟶
Canonical ordering of strings in the language

Definition: **Ø denotes the Empty language**

- Let L = {ε}; Is L=Ø? $\boxed{\text{NO}}$

The Membership Problem

Given a string $w \in \Sigma^*$ and a language L over Σ, decide whether or not $w \in L$.

<u>Example:</u>

Let w = 100011

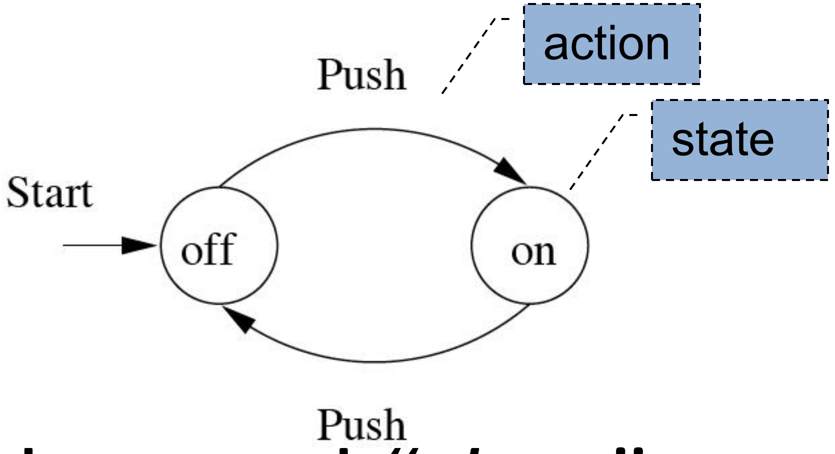
Q) Is $w \in$ the language of strings with equal number of 0s and 1s?

Finite Automata

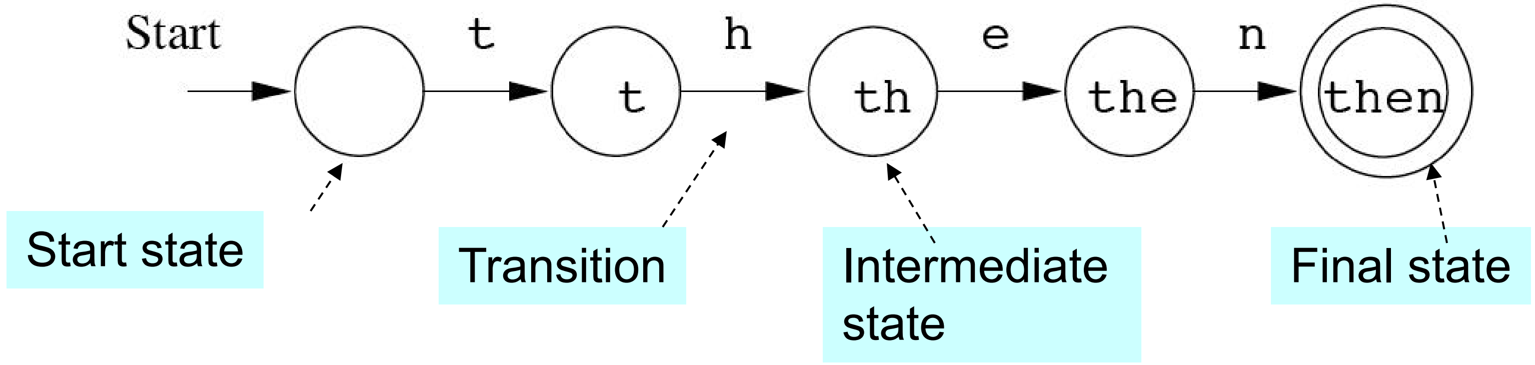
- Some Applications
  - Software for designing and checking the behavior of digital circuits
  - Lexical analyzer of a typical compiler
  - Software for scanning large bodies of text (e.g., web pages) for pattern finding
  - Software for verifying systems of all types that have a finite number of states (e.g., stock market transaction, communication/network protocol)

Finite Automata : Examples

- On/Off switch

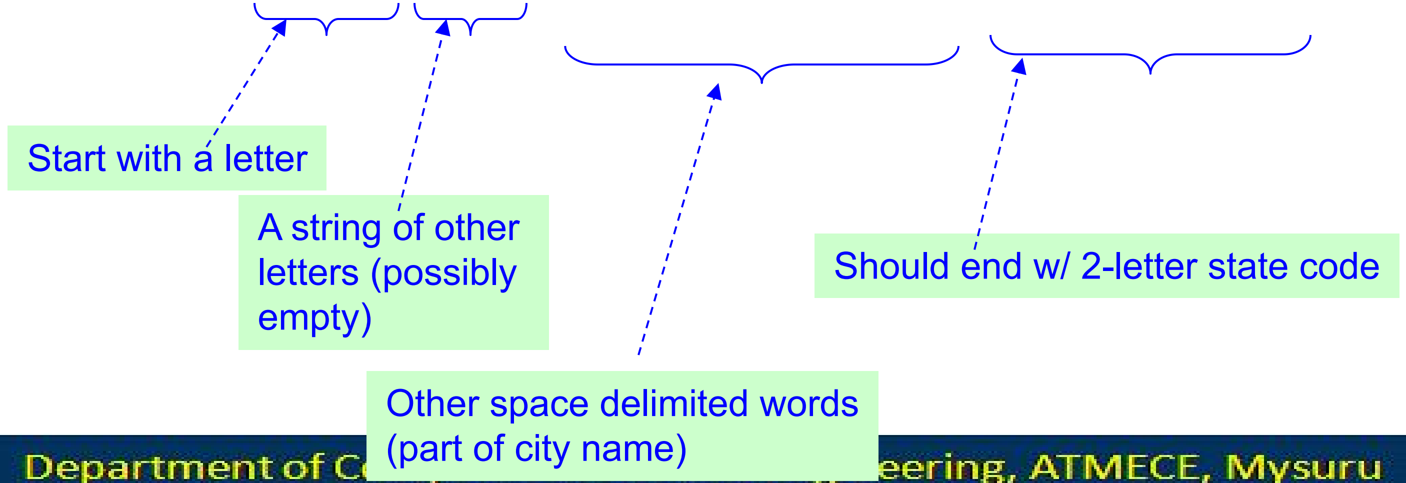


- Modeling recognition of the word "*then*"



Structural expressions

- Grammars
- Regular expressions
  - E.g., unix style to capture city names such as “Palo Alto CA”:
    - [A-Z][a-z]*([ ][A-Z][a-z]*)*[ ][A-Z][A-Z]



Start with a letter

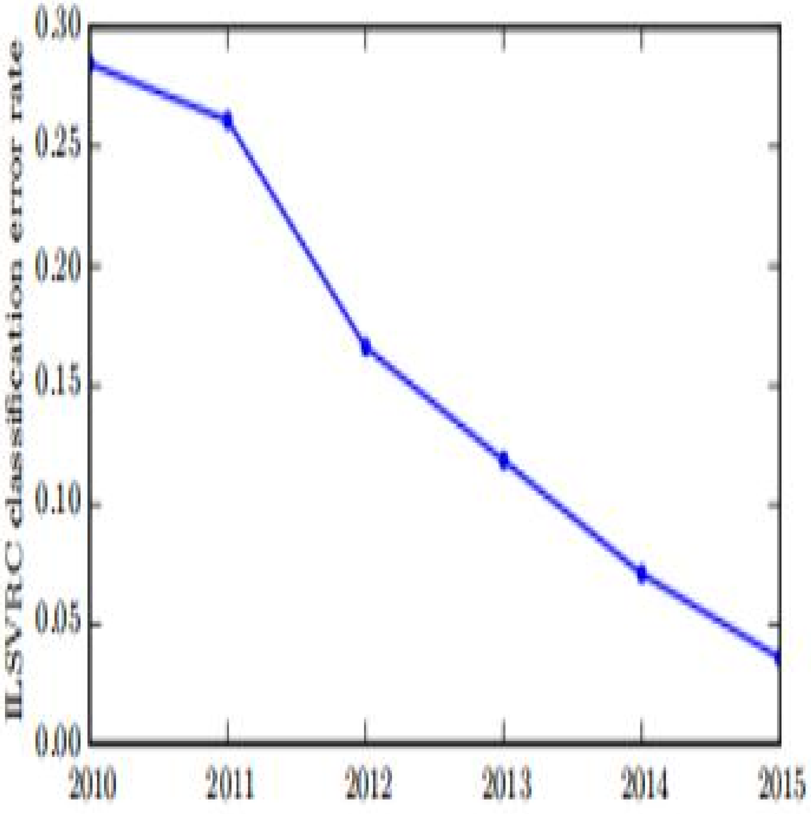
A string of other letters (possibly empty)

Should end w/ 2-letter state code

Other space delimited words (part of city name)

Increasing Accuracy, Complexity and Real-World Impact

- The earliest deep models were used to recognize individual objects in tightly cropped, extremely small images (Rumelhart et al., 1986a). Since then there has been a gradual increase in the size of images neural networks could process.
- Similarly, the earliest networks could only recognize two kinds of objects (or in some cases, the absence or presence of a single kind of object), while these modern networks typically recognize at least 1,000 different categories of objects
- The largest contest in object recognition is the **ImageNetLarge Scale Visual Recognition Challenge** (ILSVRC) held each year.
- A dramatic moment in the meteoric rise of deep learning came when a convolutional network won this challenge for the first time and by a wide margin, bringing down the state-of-the-art top-5 error rate from 26.1% to 15.3%
- DeepMind demonstrated that a reinforcement learning system based on deep learning is capable of learning to play Atari video games, reaching
- human-level performance on many tasks



Thank you