Welcome to
Theory of Computing (CSE-319)

Mohammad Hasan
CSE, CUET

Key Points

- Why Study Automata?
- What the Course is About

INTRODUCTION

- Automata theory is the study of abstract devices/machines.
- Before there were computers, in the 1930's, Alan Turing studied an abstract machine that had all the capabilities of today's computers, at least as far as in what they could compute.
- What a computing machine could do? what is could not do?

Course objectives

- What is an algorithm?
- What can and what cannot be computed?
- When should an algorithm be considered practically feasible?
- Search answer more than 70 years?
- To introduce fundamental ideas, models & results that permeate computer science

Why studying?

- Much of modern computer science is based more/less on them
- These ideas/models are powerful, beautiful, excellent examples of mathematical modeling that is elegant, productive, & of lasting value
- Besides, they are so much a part of the history & the collective subconscious, that it is hard to understand computer science without first being exposed to them

Why Study Automata?

- Finite automata are a useful model for many important kinds of hardware & software:

1. Software for designing & checking the behavior of the digital circuits
2. The lexical analyzer of a typical compiler, that is, the compiler component that breaks the input text into logical units, such as identifiers, keyword, and punctuation.
3. Software for scanning large bodies of text, such as collections of web pages, to find occurrences of words, phrases, or other patterns

Why Study Automata?

4. Software for verifying systems of all types that have a finite number of distinct states, such as communications protocols or protocols for secure exchange of information

-finite number of states

-implement in hardware as a circuit/simple form of program

-use limited amount of data/using position in the code itself to make the decision

How Could That Be?

- Regular expressions are used in many systems.
  - E.g., UNIX a.*b.
  - E.g., DTD's describe XML tags with a RE format like person (name, addr, child*).
- Finite automata model protocols, electronic circuits.
  - Theory is used in *model-checking*.

How? – (2)

- Context-free grammars are used to describe the syntax of essentially every programming language.
  - Not to forget their important role in describing natural languages.
- And DTD's taken as a whole, are really CFG's.

How? – (3)

- When developing solutions to real problems, we often confront the limitations of what software can do.
  - *Undecidable* things – no program whatever can do it.
  - *Intractable* things – there are programs, but no fast programs.
- CSE331 gives you the tools.

Other Good Stuff in CSE-331

- We'll learn how to deal formally with discrete systems.
  - Proofs: You never really prove a program correct, but you need to be thinking of why a tricky technique really works.
- We'll gain experience with abstract models and constructions.
  - Models layered software architectures.