Formal Language & Automata Theory (CS303)

Instructor: Abyayananda Maiti

Mail id: abyaym@iitp.ac.in

Phone: 8130

Syllabus

Alphabet, languages and grammars.

Regular languages and finite automata: Regular expressions and languages, deterministic finite automata (DFA), nondeterministic finite automata (NFA), regular grammars and equivalence with finite automata, properties of regular languages, pumping lemma for regular languages.

Context-free languages and pushdown automata: Context-free grammars (CFG) and languages (CFL), Chomsky and Greibach normal forms, nondeterministic pushdown automata (PDA), ambiguity in CFG, pumping lemma for context-free languages, deterministic pushdown automata.

Context-sensitive languages: Context-sensitive grammars (CSG) and languages, linear bounded automata.

Turing machines: The basic model for Turing machines (TM), Turing- recognizable (recursively enumerable) and Turing-decidable (recursive), variants of Turing machines

Undecidability: Church-Turing thesis, universal Turing machine, reduction between languages and Rice's theorem, undecidable problems about languages.

Books

1. J. E. Hopcroft, R. Motwani and J. D. Ullman, Introduction to Automata Theory, Languages and Computation, Pearson Education India (3rd edition).
2. K. L. P. Mishra, N. Chandrasekaran, Theory of Computer Science: Automata, Languages and Computation, PHI Learning Pvt. Ltd. (3rd edition).
3. D. I. A. Cohen, Introduction to Computer Theory, John Wiley & Sons, 1997.
4. J. C. Martin, Introduction to Languages and the Theory of Computation, Tata McGraw-Hill (3rd Ed.).
5. H. R. Lewis and C. H. Papadimitriou, Elements of the Theory of Computation, Prentice Hall, 1997.
6. Garey, D.S., Johnson, G., Computers and Intractability: A Guide to the Theory of NP- Completeness, Freeman, New York, 1979

Evaluation Policy

EndSem - 25%

MidSem - 25%

Class performance etc. - 10%

Regular assessments - 40%

Why Study Automata Theory and Formal Languages?

- A survey of Stanford grads 5 years out asked which of their courses did they use in their job.
- Basics like Programming took the top spots, of course.
- But among optional courses, Automata Theory stood remarkably high.
  - 3X the score for AI, for example.

Why Finite Automata and Regular Expressions?

- Regular expressions (REs) are used in many systems.
  - E.g., UNIX, Linux, OS X,... a.*b.
  - E.g., Document Type Definitions describe XML tags with a RE format like person (name, addr, child*).
- Finite automata model protocols, electronic circuits.
  - Theory is used in model-checking.

Why Context-Free Grammars?

- Context-free grammars (CFGs) are used to describe the syntax of essentially every modern programming language.

- Every modern complier uses CFG concepts to parse programs
    - Not to forget their important role in describing natural languages.

- And Document Type Definitions are really CFG's.

Why Turing Machines?

- When developing solutions to real problems, we often confront the limitations of what software can do.
  - Undecidable things – no program can do it 100% of the time with 100% accuracy.
  - Intractable things – there are programs, but no fast programs.
- A course on Automata Theory and Formal Languages gives you the tools.

Other Good Stuff

- 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.
- You'll gain experience with abstract models and constructions.
    - Models layered software architectures.

Course Outline

- Regular Languages and their descriptors:
  - Finite automata, nondeterministic finite automata, regular expressions.
  - Algorithms to decide questions about regular languages, e.g., is it empty?
  - Closure properties of regular languages.

Course Outline – (2)

- Context-free languages and their descriptors:
  - Context-free grammars, pushdown automata.
  - Decision and closure properties.

Course Outline – (3)

- Recursive and recursively enumerable languages.
  - Turing machines, decidability of problems.
  - The limit of what can be computed.

- Intractable problems.
  - Problems that (appear to) require exponential time.
  - NP-completeness and beyond.