Effective Teaching of Theory of Computation

Objective

The aim of this module is to help teachers in colleges/universities (participants) improve their teaching of the Theory of Computation (ToC) course as per the AICTE ToC course curriculum (Given in Appendix I towards the end of this document). The goal of this course is to help the participants achieve the following objectives:

- Have a clearer understanding of the importance of this course.
- Get a better understanding of the course syllabus and concepts.
- Develop meaningful course material for their courses such as assignments, exams, etc.
- Receive feedback on assignments and course resources.

Requirement for Module Participants

The participants of this module are expected to do the following:

- Taught this course at least once in their respective college/university.
- Spend at least 5 hours per week on this module.
- Regularly participate in in-class discussions and do the assignments.

Module Syllabus

The ToC module will be divided into three parts. The syllabus for each part will be as follows:

Part 1: Finite Automata and Regular Languages. (Weeks 1-2)

- Motivation for studying automata theory
- Alphabets, formal languages, and problems.
- What are regular languages and automata models for them: Deterministic Finite automaton, Formal argument of correctness, Regular languages
- Properties of regular languages-Closure, properties, product construction
- Limitations of Automata Non-regularity, Pumping Lemma
- Non-deterministic Finite Automaton, Subset construction, Equivalence with DFAs.
- Regular expressions. Equivalence with regular languages.
- Algorithms for regular languages, Minimization and its algorithm.

Part 2: Grammars and Context Free Languages. (Weeks 3-4)

- Grammars and the motivation from language theory.
- Context-free grammars, closure properties. Chomsky Normal Form for CFGs.
- PDAs. Empty-stack vs Final state acceptance conditions. Equivalence of PDAs and CFGs.
- Limitations of PDA computation, non context-free language. Pumping Lemma for CFLs.
- Deterministic CFLs and PDAs.

Part 3: Turing Machines and Computability. (Weeks 5-6)

- Modelling computation using Turing Machines. Equivalent models. Church Turing Hypothesis.
- Decidability and Turing recognizability (i.e., recursive and recursively enumerable).
- Closure properties.
- Undecidability by diagonalization.
- Reductions to show undecidability. Examples of reductions.
- Resource bounded Turing machines & Intro to Complexity. Basic complexity classes. Time bounded classes: P, NP, EXP.

Meeting Schedule

The module will be taught in an online mode. We will have a lecture once a week on Wednesdays from 4:30pm to 6:00pm. Additionally there will be a tutorial session on Saturdays from 4:30pm to 6:00pm.

References

The reference textbooks used in this module will be:

1. Introduction to the Theory of Computation, 3rd edition. Michael Sipser, Cengage Publications (Low-cost Indian edition available).
2. Introduction to Automata, Theory, Languages and Computation. Third Edition. John Hopcroft, Rajeev Motwani, Jeffrey D. Ullmann, Pearson Publications (Low-cost Indian edition available).

Appendix I (AICTE ToC Course Curriculum)

Theory of Computation Course Curriculum

Prerequisites

- Familiarity with basic data structures and algorithm design
- Must have done a Discrete Mathematics course

Essential Learning Objectives:

- Understand models and abstractions: automata as a basic model of computation
- Link between languages, automata, and decision problems.
- How to build new models from old ones: product, union, closure properties.
- Argue about limitations of computational models.
- Understand algebraic formalisms of languages such as regular expressions, context free grammars.
- Understand algorithms and computability through the lens of Turing machines.
- Existence of unsolvable problems and what that means.
- Relations between the various computational models.

Desirable Learning Outcomes

Module (appx dur in wks)	**Topics**	**Teaching Suggestions**	**Learning outcomes**

Module 1: Finite Automaton (4-5 wks)	-Why automata theory? -Alphabets, formal languages, and problems. -What are regular languages and automata models for them: Deterministic Finite automaton, Formal argument of correctness, Regular languages -Properties of regular languages-Closure, properties, product construction -Limitations of Automata Non-regularity, Pumping Lemma -Non-deterministic Finite Automaton, Subset construction, Equivalence with DFAs. -Regular expressions. Equivalence with regular languages. -Algorithms for regular languages, Minimization and its algorithm. -(suggested) Myhill-Nerode relations, Characterization of regular languages	-Sec 1.1 of T2 -Sec 1.2, 1.5 of T2 -Sec 1.1 of T1 -Sec 1.1 of T1 -Sec 1.4 of T1 -Sec 1.2 of T1 -Sec 1.3 of T1 -Sec 4.3, 4.4 of T2 -Lecture 15,16 of R1 +applications of automata to text search and NLP +applications of regular expressions for text search in UNIX. **Advanced Topics:**	F. Familiarity with notations. U. Give examples of languages, regular languages. U. Design finite automata, both deterministic and nondeterministic for a given language. R. Write formal proof of correctness of a DFA U. Give examples of non-regular languages and prove that language is non-regular using pumping lemma F. Understand the difference between determinism and nondeterminism U. Use closure properties to show non-regularity U. Design regular expressions U. Use the minimization algorithm to minimize a given DFA U. (suggested) Apply Myhill-Nerode Theorem to show that a language is regular or non-regular

		- 2DFAs, Equivalence with DFAs using Myhill-Nerode Relations (Lecture 17,18 of R1)	
Module 2: Grammars, Context-free Languages and machine models. (4-5 wks)	-Grammars and the motivation from language theory. -Context-free grammars, closure properties. Chomsky Normal Form for CFGs. -PDAs. Empty-stack vs Final state acceptance conditions. Equivalence of PDAs and CFGs. -Limitations of PDA computation, non context-free language. Pumping Lemma for CFLs. -Deterministic CFLs and PDAs. -(suggested) CYK Algorithm for parsing of CFLs.	-Sec 2.1 of T1 -Sec 2.1 of T1 -Sec 2.2 of T1 -Sec 2.3 of T1 -Sec 2.4 of T1 -Sec 7.4 of T2 +applications to parsers and compilers. **Advanced Topics:** - Ogden's Lemma.	U. Design CFGs and PDAs for CFLs R. Prove correctness of CFGs F. Understand that regular languages are a subset of CFLs. R. Prove equivalence of CFGs and PDAs U. Argue a language is non-CFL using pumping lemma F. Familiarity with DPDAs U.(suggested) Construction of DPDAs U. (suggested) Parsing using CYK algorithm

Module 3: *Turing machines and Computability,* (4-5 wks)	-Modeling computation using Turing Machines. Equivalent models. Church Turing Hypothesis. -Decidability and Turing recognizability (i.e., recursive and recursively enumerable). Closure properties. -Undecidability by diagonalization. -Reductions to show undecidability. Examples of reductions. -Resource bounded Turing machines & Intro to Complexity. Basic complexity classes. Time bounded classes: P, NP, EXP. -(suggested) Post's correspondence problem and other undecidable problems -(suggested) Polytime reductions, NP-completeness, Cook-Levin Theorem without proof	-Sec 3.1, 3.2, 3.3 of T1 -Sec 4.1 of T1 -Sec 4.2 of T1 -Sec 9.3 of T2. Sec 5.1, 5.3 of T1. -Sec 7.1 of T1 -Sec 5.2 of T1 -Sec 7.3, 7.4, 7.5 **Advanced Topics:** - Rice's Theorem - Space bounded computations and complexity, PSPACE	F. Understand relation between the various classes such as decidable, Turing recognizable., co-Turing recognizable. F. Give examples of decidable languages, undecidable languages, Turing recognizable languages. U. Prove a language is undecidable by reduction from a known undecidable problem F. Relation between basic complexity classes F. (suggested) Scenarios in which the reductions are used R. (suggested) Proving languages are NP-complete using reductions

Notations:

- Topic Categorization:

Compulsory - Topics that should be covered.

Suggested - Optional topics that the instructor can choose from given availability of time.

Advanced - Advanced topics in each module that an instructor can teach depending on the interest of the class.

- + indicates applications that could be mentioned in the class.

- Learning Outcome Categorization:
Familiarity - Student should be able to identify and comprehend *what* the topic is about. This corresponds to the cognitive levels of *knowledge* and *comprehension* of Bloom's taxonomy (see e.g., https://en.wikipedia.org/wiki/Bloom's_taxonomy).
Usability - Student should be able to understand *how* a particular idea/topic can be used, to solve problems, design examples, etc. This corresponds to the cognitive levels of *application* and *synthesis* of Bloom's taxonomy.
Reasoning - Student should have a deeper understanding of a particular concept and *why* it works. This corresponds to the cognitive levels of *analysis* and *synthesis* of Bloom's taxonomy.

Nature of lab / assignment / practice / tutorial:

1. Make assignments using the books. To test:

    - what was done in the class

    - whether the student can think and apply the concepts.

2. Tutorials: Weekly problem-solving sessions.

Suggested textbooks:

T1. Introduction to the Theory of Computation, 3rd edition. Michael Sipser, Cengage Publications (Low-cost Indian edition available).

T2. Introduction to Automata, Theory, Languages and Computation. Third Edition. John Hopcroft, Rajeev Motwani, Jeffrey D. Ullmann, Pearson Publications (Low-cost Indian edition available).

Additional Reference Material:

R1. Automata and Computability, Dexter C. Kozen. Part of the Undergraduate Texts in Computer Science book series (UTCS), Springer.

R2. Elements of the Theory of Computation, 2nd edition. Harry Lewis, Christos Papadimitriou, Prentice Hall.