Lecture 0

Course Overview

0.1 Course Description

Central to the *theory of computation* are the concepts of automata, formal languages, grammar, algorithms, computability, decidability, and complexity. Why study theory when the current focus of Computer Science (and all the more so for Information Systems) is on technology and the pragmatic areas of knowledge concerned with the development and management of computer information systems? The reasons are manifold. Theory provides a simple, elegant view of the complex machine that we call a computer. Theory possesses a high degree of permanence and stability, in contrast with the ever-changing paradigms of the technology, development, and management of computer systems. Further, parts of the theory have direct bearing on practice, such as Automata on circuit design, compiler design, and search algorithms; Formal Languages and Grammars on compiler design; and Complexity on cryptography and optimization problems in manufacturing, business, and management. Last, but not least, research-oriented students will make good use of the theory studied in this course.
This course will focus on fundamental mathematical models of computation. We will be interested in both the inherent capabilities and limitations of these computational models as well as their relationships with formal languages. Rigorous arguments and proofs of correctness will be emphasized. Enhance/develop students' ability to understand and conduct mathematical proofs for computation and algorithms. Particular topics to be covered include: (1) Finite automata, regular expressions, regular languages, and regular grammars, (2) Deterministic and nondeterministic computations on various automata, (3) Context free grammars, languages, pushdown-automata and pumping lemmas, (4) Turing machines, Church-Turing thesis, decidability, halting problem and reducibility

0.2 Course Objectives

1. Be able to design finite state machines and the equivalent regular expressions.

2. Be able to prove the equivalence of languages described by finite state machines and regular expressions.
3. Be able to design pushdown automata and the equivalent context free grammars.
4. Be able to prove the equivalence of languages described by pushdown automata and context free grammars.
5. Be able to construct Turing machines and be able to prove the equivalence of languages described by Turing machines.
6. Be able to analyse a language is decidable or undecidable and apply reductions accordingly.

0.3 Course Details

Subject Code	:	CSE 3031
Subject Name	:	`Theory of Computation`
Offered in	:	5^{th} Semester, B.Tech (CSE & CSIT)
Credits	:	4
Grading pattern	:	1
Contact Hours	:	5 Hours/Week (3 Lectures/Week, 3×1 Hour/Lecture, 1 Problem Solving Session/Week, 1×2 Hours/Problem Solving Session)

0.4 Text Book(s) and Reference(s)

Text Book:

(1) Michael Sipser, ***Introduction to the Theory of Computation***, 3^{rd} Edition, CENGAGE learning.

0.5 Lesson Plan

Contact hour	***Topics to be covered***	***Remarks (if any)***
Week #1:		

L-0	Introduction to the course and its motivation. Course description, objective, credit, grading pattern, lecture and problem solving session of the course. NBA provided program outcomes and departmental specific program specific outcomes.	**(Class: w.e.f from 23-09-2020)** Academic regulation 2018/2019
L-1	Introduction to automata, computability and complexity theory, Mathematical Notions and Terminology: Set, Sequences and tuples, Functions and Relations, Graphs, Strings and Languages, Boolean logic.	Sipser page 1-16
L-2	Definitions, Theorems, and proofs: Finding Proofs, Types of Proof: Proof by construction, proof by contradiction, proof by induction. Specifically proof $\sqrt{2}$ is an irrational number in different ways(i.e. by contradiction, by geometry etc.)	Sipser page 17-24
PSS-1	Solve as much as problems from exercise with new kind of problems related to this section	Sipser page 25$\cdots$
Week #2:		
L-3	**Regular Languages:** Finite Automata: Formal definition of a finite automaton, Examples of finite automata, formal definition of computation	Sipser page 31-37
L-4	Designing finite automata	Sipser page 37-41
L-5	Designing finite automata contd$\cdots$	Sipser page 37-41$\cdots$
PSS-2	Problems on Deterministic finite automata(Design, transition diagram, transition table, acceptance/rejection(Book Exercise: 1.1, 1.2, 1.3, 1.4, 1.5, and 1.6)	
Week #3:		
L-6	Regular operations(union, concatenation, star). The class of regular languages is closed under the union operation, and concatenation operation.	Sipser page 44
L-7	NonDeterminism: Formal definition of a nondeterministic finite automaton, NFA examples and sample design	Sipser page 44

L-8	Equivalence of NFAs and DFAs	Sipser page 54
PSS-3	Problems on regular operations(Exercise-1.4), NFAs(Exercise-1.7), NFA-DFA conversion	
Week #4:		
L-9	NFA and regular operations and introduction to regular expressions	Sipser page 63-66
L-10	Equivalence of regular expression and finite automata	Sipser page 66
L-11	Equivalence of regular expression and finite automata contd···	Sipser page 66
PSS-4	problems: Conversion of regular expression to nondeterministic finite automata, finite automata to regular expression	
Week #5:		
L-12	Non-regular languages: The pumping lemma for regular languages, proof, pigeonhoe principle	Sipser page 77
L-13	Examples on pumping lemma	Sipser page 77
L-14	Examples on pumping lemma contd··· & more discussion on closure properties of regular sets	Sipser page 77
PSS-5	Discuss problems related to pumping lemma(Exercise-1.29, 1.49, 1.51 etc)	Sipser page 82 ···
Week #6:		
L-15	**Context-Free Languages:** Context-Free Grammars: Formal Definition of a context-free grammars, Examples of context-free grammars, Designing context-free grammars	Sipser section 2.1 (page 101-106)
L-16	Ambiguity, and Chomsky normal form	Sipser (page 107-108)
L-17	Introduction to pushdown automata(PDA): formal definition of a pushdown automata, Examples on pushdown automata	Sipser (page(111-113)

PSS-6	Problems context-free grammar, pushdown automata, Chomsky normal form	Sipser (page 154-156)
Week #7:		
L-18	Pushdown automata and Equivalence with context-free languages	Sipser page 117
L-19	Non context-free languages: The pumping lemma for context-free languages	Sipser page 125
L-20	Non context-free languages: The pumping lemma for context-free languages $\cdots$	Sipser page 125
PSS-7	Problems on equivalence between context-free languages and pushdown automata, pumping lemma for context-free languages	
Week #8:		
L-21	Deterministic context-free languages(DCFL): Properties of DCFLs	Sipser page 130-133
L-22	Deterministic context-free grammars, Relationship of DeterministicPDAs and DCFGs	Sipser page 135-146
L-23	Deterministic context-free grammars, Relationship of DPDAs and DCFGs $\cdots$, Parsing and LR(k) grammars	Sipser page 146-151
PSS-8	Problems on DPDA, DCFL	
Week #9:		
L-24	**Computability Theory**:Turing Machines: Formal definition of a Turing machine, Examples of Turing machine	Sipser page 165-167-170
L-25	Examples of Turing machine contd$\cdots$	Sipser page 167-170
L-26	Variants of Turing machines: Multitape Turing machines, Nondeterministic Turing machine	Sipser page 176-178
PSS-9	Problems and discussion on multitape Turing machine	
Week #10:		

L-27	Enumerators, equivalence with other models	Sipser page 180-181
L-28	Variants of Turing machines*cdots*	Sipser page 176-181
L-29	The Definition of Algorithm: Hilbert's Problem	Sipser Sec 3.3 page 182
PSS-10	Problems to be discussed related to Sipser Sec 3.2, and 3.3	
Week #11:		
L-30	**Decidability:** Decidable Languages, decidable problems concerning regular languages	Sipser Chapter 4 page 193-194
L-31	Decidable problems concerning context-free languages	Sipser page 194-198
L-32	Decidable problems$\cdots$	Sipser page 194-198
PSS-11	Problems and Discussions related to decidability	Sipser page 210
Week #12:		
L-33	**Undecidability:**The diagonalization method	Sipser page 201-202
L-34	An undecidable language	Sipser page 207
L-35	A Turing-unrecognizable language	Sipser page 207-209
PSS-12	Problems and Discussions on undicidability	Sipser page 210$\cdots$
Week #13:		
L-36	**Reducibility:**undecidable problems from language theory	Sipser page 215-216
L-37	Reduction via computation histories	Sipser page 216-220

L-38	Reduction via computation histories· · ·	Sipser page 216-220
PSS-13	A Simple reduction problems, Problems related to reductions	Sipser page 220-227
Week #14:		
L-39	Mapping reducibility	Sipser page 234
L-40	Computable functions	Sipser page 234
L-41	Formal definition of mapping reducibility	Sipser page 235
PSS-14	Problems and discussions on reducibility	Sipser page 239 . . .

0.6 Course Outcomes

By the end of course through lectures, problem solving sessions, and exams students will be able to:

CO 1. enhance/develop ability to understand and conduct mathematical proofs for computation and algorithms.

CO 2. design and analyze finite automata, and regular expression for describing regular languages.

CO 3. design and analyze pushdown automata, and context-free grammars.

CO 4. design and analyze Turing machines.

CO 5. design, implement, and evaluate computational models to meet desired needs of the languages, and formulate computational models for real-life problems.

CO 6. demonstrate the understanding of key notions, such as algorithm, computability, decidability, and complexity through problem solving.

0.7 Program Outcomes & Program Specific Outcomes

0.7.1 Program Outcomes (POs)

There are twelve program outcomes (1-12) for the Computer science & Engineering B. Tech program:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

0.7.2 Program Specifis Outcomes (PSOs)

PSO 1. The ability to understand, analyze and develop computer programs in the areas related to business intelligence, web design and networking for efficient design of computer-based systems of varying complexities.

PSO 2. The ability to apply standard practices and strategies in software development using open-ended programming environments to deliver a quality product for business success.