COURSEPACK (Fall 2024-25)

1. THE SCHEME

Course Title	Theory of Computation	Course Type	Theory
Course Code	R1UC501T	Class	B.Tech CSE and specializations V Sem

Instruction delivery	Activity	Credits	Credit Hours	Total Number of Classes per Semester				Assessment in Weightage	
	Lecture	3	3	Theory	Tutorial	Practical	Self-study	CIE	SEE
	Tutorial	0	0						
	Practical	0	0						
	Self-study	0	6						
	Total	3	3	45	0	0	6	50%	50%

Course Lead	Munish Khanna	Course Coordinator	Avaneesh Kumar Yadav

Names Course Instructors	Theory
	1. Munish Khanna 2. Avaneesh Kumar Yadav 3. Sanjeev Kumar Punia 4. Arvind Dagur 5. Arun Kumar Rai 6. Nand Kumar Jyotish 7. Pradeep Bedi 8. Sunil Kumar(33043) 9. P Sudhakar 10. Rishav Raj 11. Braj Mohan Singh 12. Sheo Kumar 13. V. Gokul Rajan 14. Aanjey Mani Tripathi 15. Arunendra Mani Tripathi 16. Tarun Kumar 17. Akhilesh Kumar Singh (32563) 18. Amit Yadav 19. Namrata Kumari 20. Prashant Dixit 21. Greeshma G.S. 22. Sunder R 23. Manmohan Singh 24. Gaurav Vinchurkar 25. Sivakumar Madeshwaran

2. COURSE OVERVIEW

This course on the Theory of Automata and Formal Languages introduces students to the foundational concepts and techniques used to model and analyze computational systems. Students will explore different types of formal languages and automata, including finite automata, pushdown automata, and Turing machines, as well as the grammars that generate these languages. Key topics include the classification of languages into regular, context-free, and recursively enumerable categories, the theory of computation, and the limits of what can be computed. Through this subject, students will develop a deeper understanding of how formal languages are used to describe and manipulate computational processes, which is essential for fields such as compiler construction, software development, and algorithm design.

3. COURSE OBJECTIVES

The objective of this course is to provide students with a comprehensive understanding of the theory of automata and formal languages. Students will learn to analyze and design various computational models, such as finite automata, regular expressions, context-free grammars, and Turing machines. The course will cover key concepts including regular languages, context-free languages, decidability, and advanced topics. By the end of the course, students will be equipped to apply these theoretical foundations to solve problems in computer science and to classify machines by their power to recognize languages and comprehend the hierarchy of problems.

4. PREREQUISITE COURSE

PREREQUISITE COURSE REQUIRED	YES	
If, yes please fill in the Details	Course code	Course Title
	C1UC224T	Discrete Mathematics

5. PROGRAM OUTCOMES (POs):

PO No.	Description of the Program Outcome
PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex Computer Science and engineering problems.
PO2	**Problem Analysis**: Identify, formulate, review research literature, and analyze complex Computer Science and engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/Development of Solutions:** Design solutions for complex Computer Science and 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.
PO4	**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.
PO5	**Modern Tool Usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex computer science and engineering activities with an understanding of the limitations.
PO6	**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.

PO7	**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.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex Computer Science and 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.
PO11	**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.
PO12	**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 changes in the field of Computer Science.

6. PROGRAM SPECIFIC OUTCOMES (PSOs):

Program Specific Outcomes (PSO) are statements that describe what the graduates of a discipline-specific program should be able to do. Two to Three PSOs per program should be designed.

PO No.	Description of the Program-Specific Outcome
PSO1	Have the ability to work with emerging technologies in computing requisite to Industry 4.0.
PSO2	Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

7. COURSE CONTENT (THEORY + PRACTICAL)

CONTENT (Syllabus)

THEORY:

FINITE AUTOMATA (FA): Introduction, Proof Techniques, Deterministic Finite Automata (DFA) - Formal definition, simpler notations (state transition diagram, transition table), language of a DFA. Nondeterministic Finite Automata (NFA)- Definition of NFA, language of an NFA, Equivalence of Deterministic and Nondeterministic Finite Automata, Applications of Finite Automata, Finite Automata with Epsilon Transitions, Eliminating Epsilon transitions, Minimization of Deterministic Finite Automata, Finite automata with output (Moore and Mealy machines)

REGULAR EXPRESSIONS (RE): Introduction, Identities of Regular Expressions, Finite Automata and Regular Expressions- Converting from DFA's to Regular Expressions, Converting Regular Expressions to Automata, applications of Regular Expressions.
REGULAR GRAMMARS: Definition, regular grammars and FA, FA for regular grammar, Regular grammar for FA. Proving languages to be non-regular -Pumping lemma, applications, and Closure properties of regular languages.

CONTEXT FREE GRAMMER (CFG): Derivation Trees, Sentential Forms, Rightmost and Leftmost derivations of Strings. Ambiguity in CFG's, Minimization of CFG's, CNF, GNF, Pumping Lemma for CFL's, Enumeration of Properties of CFL (Proof's omitted). **PUSHDOWN AUTOMATA:** Definition, Model, Acceptance of CFL, Acceptance by Final State and Acceptance by Empty stack and its Equivalence, Equivalence of CFG and PDA. TURING MACHINES (TM): Formal definition and behaviour, Languages of a TM, TM as accepters, and TM as a computer of integer functions, Types of TMs. **RECURSIVE(RE) AND RECURSIVELY ENUMERABLE LANGUAGES (REL):** Properties of recursive and recursively enumerable languages, Universal Turing machine, The Halting problem, Undecidable problems about TMs. Context sensitive language and linear bounded automata (LBA), Chomsky hierarchy, Decidability, Post's correspondence problem (PCP), un-decidability of PCP.

8. COURSE OUTCOMES (COs)

After the completion of the course, the student will be able to:

CO No.	Description of the Course Outcome
R1UC501T.1	Mathematical background required to understand the subject.
R1UC501T.2	Construct regular expressions for different languages and create Deterministic Finite Automata (DFA) and Non-Deterministic Finite Automata Machine (NFA).
R1UC501T.3	Model Push Down Automata (PDA) for Context Free Languages (CFLs).
R1UC501T.4	Design Turing Machine for various recursive enumerable languages. Understanding the basics of Undecidability and Halting problem.

9. TAXONOMY LEVEL OF THE COURSE OUTCOMES

Mapping of COs with Bloom's Level

CO No.	Remember **KL1**	Understand **KL 2**	Apply **KL 3**	Analyse **KL 4**	Evaluate **KL 2**	Create **KL 6**
R1UC501C.1			√			
R1UC501C.2			√	√		
R1UC501C.3			√	√		
R1UC501C.4			√			√

10. COURSE ARTICULATION MATRIX

COs#/ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
R1UC501C.1	2	1	1											
R1UC501C.2	2	2	1											
R1UC501C.3	2	2	2											
R1UC501C.4	2	2	2		1							1		

Note: 1-Low, 2-Medium, 3-High \ *first semester first course and first Course Outcome

11. TYPICAL EXAMPLE OF COURSES, CREDIT HOURS AND TEACHING HOURS

Type of Course (T)	**CIE**			**Total Marks**		**Final Marks CIE*0.5+SEE*0.5**
	IA1#	MTE	IA2#	CIE	SEE	
THEORY	25	50	25	100	100	100

*1 credit = 3 self-learning hours (Not to mention in the lesson plan)

12. TYPICAL EXAMPLE OF COURSES, CREDIT HOURS AND TEACHING HOURS

Type of Course	**Credits Hours**					**Hours of engagement/ Week**					**12 weeks/ semester**	**Remarks**
	Theory	**Tutorial**	**Practical**	**Self-study**	**Total**	**Theory**	**Tutorial**	**Practical**	**Self-study**	**Total**	**Total no. of classes**	
Theory Course	3	0	0	0	3	3	0	0	0	3	40	40 classes for theory

*1 credit = 3 self-learning hours (Not to mention in the lesson plan)

LESSON PLAN FOR THEORY COURSES (THEORY AND TUTORIAL CLASSES) FOR THEORY
15 weeks * 3 Hours = 45 Classes (1credit = 1 Lecture Hour)

L. No.	Topic for Delivery	Tutorial/Practical Plan	Skill	Competency
1.	Introduction, Background on Sets, Relations and Graphs	Theory	Write and understand mathematical background	CO1
2.	Background on different types of mathematical proofs: Deductive, Contradiction, Induction, and Contra positive.			
3.	Formal Definition of Deterministic Finite Automata (DFA)	Theory	Formal definition of DFA	CO2
4.	State transition diagram,Examples of languages accepted by DFA			
5.	Problem solving on DFA			
6.	Problem solving on DFA			
7.	NFA: Formal definition		Construction of state transition diagram; Formal definition of NFA	
8.	Examples of languages accepted by NFA			
9.	Problem solving on NFA			
10.	Minimization of DFA and Myhill Nerode theorem	Theory	Construct equivalent automata with possibly fewer states	
11.	Problem solving on Minimization of DFA	Theory	Inter conversion between finite automata	
12.	Empty moves, conversion of NFA with empty moves into NFA without empty moves			
13.	Conversion of NFA without empty moves into DFA			
14.	Finite Automata with outputs (Mealy machine and Moore Machine)	Theory	Constructing automata with output	
15.	Problem solving on Mealy machine and Moore Machine			
16.	Inter conversion between Mealy machine and Moore machine			
17.	Introduction to Regular Expressions (RE), Properties of RE	Theory	Conversion from regular expression to DFA	
18.	Problem solving on RE			
19.	DFA to RE and RE to DFA			
20.	Closure Properties of Regular Languages (RL)	Theory	Analyze properties of regular languages	
21.	Pumping lemma for RL			
22.	Problem solving on pumping lemma for RL			
23.	Introduction to Grammar and its four tuples			
24.	Introduction to Context Free Grammar (CFG) and Problem solving on generating language from given grammar.	Theory	Derivations, Parse trees, and applications for CFG	
25.	Numerical on creation of grammar for the given language			
26.	Derivation and Derivation Tree			
27.	Left most derivation and Right Most Derivation			
28.	Discussion on Ambiguity and removal of ambiguity			

29.	Elimination of epsilon moves, Removal of useless production, removal of unit production and removal of left recursion from grammar	Theory	Conversion to normal form	CO3
30.	Numerical on Elimination of epsilon moves, Removal of useless production and removal of unit production			
31.	Conversion of the grammar into Chomsky Normal Form (CNF)			
32.	Conversion of the grammar into Greibach Normal Form (GNF)			
33.	Introduction to Pushdown automata (PDA) and its type (Deterministic PDA (DPDA) and Non-deterministic PDA(NPDA))	Theory	Definition and designing PDA	
34.	Numerical on NPDA and DPDA			
35.	Numerical on NPDA and DPDA and introduction to 2-PDA			
36.	PDA to CFG			
37.	CFG to PDA			
38.	Properties of Context Free Languages (CFL)			
39.	Pumping lemma for CFL	Theory	Deciding which languages are CFL or not	
40.	Chomsky hierarchy of languages (Focusing on Regular Languages, Context Sensitive Languages)	Theory	Discussion on different classes of languages	
41.	Turing Machine and its variants (Like Linear Bounded Automata and Universal Turing Machine)	Theory	To formally describe computation by a Turing Machine; To recognize undecidable languages; Analysis of complexity of languages	CO4
42.	Numerical on Turing Machine			
43.	Halting problem and introduction to Undecidability			
44.	Recursive (REC) Languagesand Recursive Enumerable (RE) Languages			
45.	Post Correspondence Problem (PCP)			

12. BIBLIOGRAPHY

Text Book
Introduction to Automata Theory, Languages and Computation 3rd Edition by John E Hopcroft, Rajeev Motwani and Jeffrey D. Ullman [Availaible Online]
Reference Books
Introduction to the Theory of Computation Second Edition by Michael Sipser [Availaible Online]
Introduction to Formal Languages, Automata Theory and Computation by Kamala Krithivasan and Rama R

Journals/Magazines/Govt. Reports/Gazette/Industry Trends
https://theory.report
https://thmatters.wordpress.com/tcs-blogs/
ALGORITHMICA, Springer Publications [Journal]
THEORY OF COMPUTATION, Elsevier [Journal]

Webliography (Two electronic documents or websites that relate to the Course)
https://math.mit.edu/~sipser/18404/

https://www.math.ias.edu/avi/book
https://cs-people.bu.edu/mbun/courses/332_F21/

SWAYAM/NPTEL/MOOCs Certification
(One from Each Platform, Max 3 Platforms) https://onlinecourses.nptel.ac.in/noc21_cs83/preview

13. COURSE ASSESSMENT

Assessment forms an integral part of curriculum design. A learning-teaching system can only be effective if the student's learning is measured at various stages which means while the student processes learning (Assessment for Learning) a given content and after completely learning a defined content (Assessment of Learning). Assessment for learning is referred to as formative assessment, that is, an assessment designed to inform instruction.

The ability to use and apply the knowledge in different ways may not be the focus of the assessment. With regard to designing assessments, the faculty members must be willing to put in the time required to create a valid, reliable assessment, that ideally would allow students to demonstrate their understanding of the information while remaining. The following are the five main areas that assessment reporting should cover.

1. **Learning Outcomes**: At the completion of a program, students are expected to know their knowledge, skills, and attitude. Depending on whether it is a UG or PG program, the level of sophistication may be different. There should be no strict rule on the number of outcomes to be achieved, but the list should be reasonable, and well-organized.
2. **Assessable Outcomes**: After a given learning activity, the statements should specify what students can do to demonstrate. Criteria for demonstration are usually addressed in rubrics and there should be specific examples of work that doesn't meet expectations, meets expectations, and exceeds expectations. One of the main challenges is faculty communication whether all faculty agreed on explicit criteria for assessing each outcome. This can be a difficult accomplishment when multiple sections of a course are taught or different faculty members. Hence there is a need for common understanding among the faculty on what is assessed and how it is assessed.
3. **Assessment Alignment**: This design of an assessment is sometimes in the form of a curriculum map, which can be created in something as easy as an Excel spreadsheet. Courses should be examined to see which program outcomes they support, and if the outcome is assessed within the course. After completion, program outcomes should be mapped to multiple courses within the program.
4. **Assessment Planning**: Faculty members need to have a specific plan in place for assessing each outcome. Outcomes don't need to be assessed every year, but faculty should plan to review the assessment data over a reasonable period of time and develop a course of action if the outcome is not being met.
5. **Student Experience**: Students in a program should be fully aware of the expectations of the program. The program outcomes are aligned on the syllabus so that students are aware of what course outcomes they are required to meet, and how the program outcomes are supported. Assessment documents should clearly communicate what is being done with the data results and how it is contributing to the improvement of the program and curriculum.

Designing quality assessment tools or tasks involves multiple considerations if it is to be fit for purpose. The set of assessments in a course should be planned to provide students with the opportunity to learn as they engage with formative tasks as well as the opportunity to demonstrate their learning through summative tasks. Encouraging the student through the use of realistic, authentic experiences is an exciting challenge for the course faculty team, who are responsible for the review and quality enhancements to assessment

15. PASSING STANDARDS

Passing Criteria for Different Course Types Effective from AY 2022-23 Onwards

S.No.	Course Type	Passing Criterion

1.	**Theory Course (T)**	A student shall secure a minimum of **30% of the maximum marks** in the semester-end examination (SEE/ETE) and **40% of aggregate marks** in the course including Continuous internal examination (CIE) and SEE/ETE marks. i.e., the minimum Passing Grade is "P".

Note: Students unable to meet the overall passing criteria as mentioned shall be eligible for the following options to clear the course:

- Appear in the Back Paper Examinations and have to meet the criteria to score 40% in marks overall
- Appear in summer examinations (Internal +External) to meet the criteria as mentioned.

16. PROBLEM-BASED LEARNING/CASE STUDIES/CLINICS

Exercises in Problem-based Learning (Assignments)

S.No.	Problem
1	Prove that the sum of first n natural numbers is equal to n*(n+1)/2
2	L1={aa, ab, aab, aba} L2={bb, aa, ba, ab, bab} find L1 - L2, L2 – L1.
3	Find SUFIX and PREFIX of the string "GALGOTIASUNIVERSITY".
4	Explain proof by construction, proof by contradiction and proof by induction through examples.
5	Design DFA for the following language over input alphabet (a,b):L = String doesn't start with aab.
6	Design DFA for the following language over input alphabet (a,b): L = Starting with a and end with b.
7	Design DFA for the following Language over input alphabet(0,1): L = Even no. of 0's or Even no. of 1's.
8	Design Mealy Machine to convert 2's Complement of the binary input.
9	How do you remove epsilon transitions from an NFA?
10	Prove that regular languages are closed under union and intersection.
.11	Design DFA for the following Language over input alphabet (0,1):L = Starting with 01 and end with 10.
12	Design DFA where string does not end with 001.
13	Give formal description of Pumping Lemma for Regular Languages.
14	Define – Moore machine. Design the mealy and moore machine to count the number of substring"ab". Design the mealy and moore machine to print 'A','B','C', depends upon the inputs that end with '10', '01' or other,
15	Given a CFL. How would you construct a PDA for it?
16	Explain derivation of a CFL sentence from a CFG through example.
17	What are Type 0, Type 1 and Type 2 languages?

18	S –> aSS / aSaS/ aSab | b find left factoring for the given grammar.
19	Apply Pumping Lemma for CFL to show a certain language is not CFL
20	What is ambiguous grammar? Explain through examples
21	Check whether string W ϵ L(G) or not using membership algorithm. W = baab S –> AB A –> BB | a B –> AB | b
22	Convert Context Free Grammar to GNF (Greibach normal form) . S –> CB / AB A –> a / AA B –> b C –> d
23	Write Context Free Grammar for the following languages : L={a^nb^n | n>=1} and L={a^mb^n | m=2n, n>=0}
24	What are recursive enumerable and recursive language?
25	Prove equivalence between PDA with two stacks and TMs
26	Explain Church Turing Thesis.
27	What are decidable and undecidable languages? Give Examples
28	Design a PDA, a to accept L = { $a^{2n}b^n$ | $n \geq 1$ }
29	Write Context Free Grammar for the language L={a^mb^n | m=2n, n>=0}.
30	Construct the DPDA Machine for language L={$a^mb^nc^m$ | m, n>=0}.
31	Prove that the following language is ambiguous and convert into
32	unambiguous S → S + S | S * S | a | b Where W = a + a * b.
33	Design a Turing Machine to convert the Binary value to 2's Complement.
34	Recursive Enumerable Languages are Decidable in case of Emptiness, Finiteness and
35	Equivalence. TRUE / FALSE. Justify your answer.
36	Construct the Turing Machine to implement adder for unary value.
37	What is PCP problem? Is it decidable or undecidable?
38	Prove that PCP is undecidable?
39	Differentiate between Decidable and Undecidable problem.

40	Identify the language L= {a^x : where x is a prime number}.
41	Design the Turing machine for Addition, proper subtraction and multiplication of two numbers
42	Design the NPDA and DPDA for L={$a^n b^n$|n>0}
43	Justify why NPDAs are more powerful than DPDA
44	Design the 2-PDA for L={$a^n b^n c^n$|n>0}
45	Discuss universal Turing machine, Linear bounded automata, and instantaneous description of TM and PDA
46	Design the PDA for L={$a^3 b^n c^n$|n>0}
47	Write the regular expression for the language starting with a but not having consecutive b's.
48	Write the regular expression for the language having a string which should have atleast one 0 and alteast one 1.
49	Write the regular expression for the language L over $\sum = \{0, 1\}$ such that all the string do not contain the substring 01.
50	Write the regular expression for the language containing the string in which every 0 is immediately followed by 11.