National University of Engineering (UNI)
School of Artificial Intelligence
Syllabus 2027-I

1. COURSE

CS211. Theory of Computation (Mandatory)

2. GENERAL INFORMATION

2.1 Course	:	CS211. Theory of Computation
2.2 Semester	:	4th Semester.
2.3 Credits	:	4
2.4 Hours	:	2 HT; 4 HP;
2.5 Duration of the period	:	16 weeks
2.6 Type of course	:	Mandatory
2.7 Learning modality	:	Face to face
2.8 Prerrequisites	:	CS1D1. Discrete Structures. (2nd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Theory of Computation provides the mathematical foundations to understand what can be computed and how efficiently. It introduces formal languages, automata, and the limits of algorithmic solvability. This knowledge is fundamental for understanding compiler construction, formal verification, and the inherent complexity of computational problems.

5. GOALS

- Master the concepts of finite automata and regular expressions.
- Analyze context-free grammars and pushdown automata.
- Understand the Universal Turing Machine as a model of computation.
- Comprehend the boundaries of computability and the Halting Problem.
- Differentiate between complexity classes like P and NP.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Formal Languages and Automata (18 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal automata: – Finite State – Pushdown – Linear Bounded – Turing Machine • Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0)) • Deterministic and nondeterministic automata • Pumping Lemma proofs: – Proof of Finite State/Regular-Language limitation – Pushdown Automata/Context-Free-Language limitation	• For each formal automaton in this unit: – Explain its definition comparing its characteristics with this unit's other automata – Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input – Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] • Given a problem, develop an appropriate automaton that addresses the problem [Create] • Develop a regular expression for a given regular language expressed in natural language [Create] • Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply] • Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply]
Readings : [**Sipser2012**], [**Hopcroft2013**]	

Unit 2: Formal Languages and Automata (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal automata: – Finite State – Pushdown – Linear Bounded – Turing Machine • Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0) • Normal Forms: Chomsky and Greibach • Pumping Lemma proofs: – Proof of Finite State/Regular-Language limitation – Pushdown Automata/Context-Free-Language limitation • Relations among formal automata, languages, and grammars	• For each formal automaton in this unit: – Explain its definition comparing its characteristics with this unit's other automata – Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input – Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] • Given a problem, develop an appropriate automaton that addresses the problem [Create] • Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply] • Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply] • Construct CFGs for formal language descriptions [Evaluar]
Readings : [**Sipser2012**], [**Hopcroft2013**]	

Unit 3: Formal Languages and Automata (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal automata: – Finite State – Pushdown – Linear Bounded – Turing Machine • Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0) • Decidability, (un)computability, and halting • The Church-Turing thesis • Reducibility and reductions • Equivalent models of algorithmic computation: – Turing Machines and Variations (e.g., multi-tape, non-deterministic) – Lambda Calculus – Mu-Recursive Functions	• For each formal automaton in this unit: – Explain its definition comparing its characteristics with this unit's other automata – Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input – Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] • Explain a universal Turing Machine and its operation [Explain] • Present to an audience of co-workers and managers the impossibility of providing them a program that checks all other programs, including some seemingly simple ones, for infinite loops including an explanation of the Halting problem, why it has no algorithmic solution, and its significance for real-world algorithmic computation [Explain] • Explain the Church-Turing Thesis and its significance for algorithmic computation [Explain] • Apply arithmetization and diagonalization to prove the Halting Problem for Turing Machines is Undecidability [Apply] • Given a known undecidable language, apply a mapping reduction or computational history to prove that another language is undecidable [Apply]
Readings : [**Sipser2012**], [**Kozen2006**]	

Unit 4: Formal Languages and Automata (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Formal languages, grammars and Chomsky Hierarchy: – Regular (Type-3): ∗ Regular Expressions – Context-Free (Type-2) – Context-Sensitive (Type-1) – Recursively Enumerable (Type-0) • Relations among formal automata, languages, and grammars • Equivalent models of algorithmic computation: – Turing Machines and Variations (e.g., multi-tape, non-deterministic) – Lambda Calculus – Mu-Recursive Functions • Context-Sensitive Languages (Type-1) and applications	• For each formal model in this unit: – Explain its definition comparing its characteristics with the others in this unit – Explain example inputs that are and cannot be accepted by the language/grammar. [Explain] • Explain the Primitive and General Recursive functions (zero, successor, selection, primitive recursion, composition, and Mu), their significance, and Turing Machine implementations [Explain] • Explain how computation is performed in Lambda Calculus (e.g., Alpha conversion and Beta reduction) [Explain] • Explain Rice's theorem and its significance [Explain]
Readings : [**Sipser2012**], [**Kozen2006**]	

Unit 5: Complexity Analysis Framework (3 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Complexity Analysis Framework: – Best, average, and worst-case performance of an algorithm – Empirical and relative (Order of Growth) measurements – Input size and primitive operations – Time and space efficiency	• Prepare a presentation that explains to first year students the basic concepts of algorithmic complexity including best, average, and worst-case algorithm behavior, Big-O, Omega, and Theta notations, complexity classes, time and space tradeoffs, empirical measurement, and impact on practical problems [Explain] • Informally evaluate the foundational complexity class of simple algorithms [Evaluate]
Readings : [**Sipser2012**], [**Cormen2022**]	

Unit 6: Asymptotic Notation and Complexity Classes (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Asymptotic complexity analysis (average and worst-case bounds): – Big-O, Big-Omega, and Big-Theta formal notations – Foundational Complexity Classes and Representative Examples/Problems: ∗ $O(1)$ Constant (e.g., array access) ∗ $O(\log_2 n)$ Logarithmic (e.g., binary search) ∗ $O(n)$ Linear (e.g., linear search) ∗ $O(n \log_2 n)$ Log Linear (e.g., mergesort) ∗ $O(n^2)$ Quadratic (e.g., selection sort) ∗ $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) ∗ $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) ∗ $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)	• Using examples, explain each of the foundational complexity classes in this unit [Explain] • Explain to a non-technical audience the significance of tractable versus intractable algorithms using an intuitive explanation of Big-O complexity [Explain]
Readings : [**Sipser2012**], [**Cormen2022**]	

Unit 7: Complexity Analysis: Recursion, Amortization, and Tight Bounds (7 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	**Learning Outcomes**
• Little-o, Little-Omega, and Little Theta notations • Formal recursive analysis	• Use recurrence relations to evaluate the time complexity of recursively defined algorithms [Apply] • Apply elementary recurrence relations using a form of the Master Theorem [Apply]
Readings : [**Sipser2012**], [**Cormen2022**]	

Unit 8: Computational Complexity Theory (6 hours)	
Competences Expected:	
Topics	**Learning Outcomes**
• Tractability and intractability: – P, NP, and NP-Complete Complexity Classes – NP-Complete Problems (e.g., SAT, Knapsack, TSP) – Reductions • Turing Machine-based models of complexity: – Time complexity: ∗ P, NP, NP-C, and EXP classes ∗ Cook-Levin theorem – Space Complexity: ∗ NSpace and PSpace ∗ Savitch's theorem	• Explain the significance of NP-Completeness [Explain] • Explain how NP-Hard is a lower bound and NP is an upper bound for NP-Completeness [Explain] • Explain examples of NP-complete problems [Explain] • Explain the Cook-Levin Theorem and the NP-Completeness of SAT [Explain] • Explain the classes P and NP [Explain] • Prove that a problem is NP-Complete by reducing a classic known NP-C problem to it (e.g., 3SAT and Clique) [Create] • Explain the P-space class and its relation to the EXP class [Explain]
Readings : [**Sipser2012**], [**Cormen2022**]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

10. BASIC BIBLIOGRAPHY