**Approval: 10th Senate Meeting**

**Course Name:** Formal Languages and Automata Theory
**Course Number:** CS-304
**Credits:** 3-0-0-3
**Prerequisites:** CS-202, CS-208 or instructor's consent
**Intended for:** B.Tech.
**Distribution:** Discipline elective for CS; CS elective for EE, ME, and Civil
**Semester:** 5th
**Preamble:** The proposed new curriculum for CS includes six discipline core courses:

1. Mathematical Foundations of Computer Science
2. Advanced Data Structures and Algorithms
3. Paradigms of Programming
4. Computer Organization
5. Information Systems
6. Communicating Distributed Processes

The proposed course follows the new CS curriculum design approach that strives to cover in the above-mentioned six core courses all the fundamental concepts that a CS undergraduate student must know of and provide an exposure to more specialized topics in various discipline electives.

This course proposal attempts to include the fundamental computation models and abstractions. The topics form the essential core of this course that must be covered comprehensively, with lots of examples and practice exercises, with emphasis on clarifying concepts and being able to reason on them. In the proposed contents, there are some topics that are marked "advanced". These topics can be introduced depending on background and interests of the class and if the time permits. These and some other advanced topics should be covered in greater detail in advanced discipline electives.

**Objective:** After the students have gone through a course on discrete structures, such as CS-208, where they learn the formal and abstract representations of data and its manipulation, and a course on data structures and algorithms, such as CS-202, where they learn concrete implementations and manipulations of such discrete structures and their use in design and analysis of non-trivial algorithms for a given computational task, this introductory course on formal languages and automata theory familiarizes the students to formal models and abstractions of computation that have been developed over the last few decades and helps them to develop an ability to form abstractions of their own and reasons in terms of them. On completion of such a course, the students should be able to

*Knowledge and Understanding:*

- Demonstrate familiarity with and manipulate the different concepts in automata theory and formal languages such as formal proofs, (non-)deterministic automata, regular expressions, regular languages, context-free grammars, context-free languages, Turing machines, and notions of computability and undecidability.
- Explain the power and the limitations of various models of computation.

*Skills and abilities:*

- Prove properties of languages, grammars and automata with rigorously formal mathematical methods.

- Design automata, regular expressions and context-free grammars accepting or generating a certain language.
- Describe the language accepted by an automata or generated by a regular expression or a context-free grammar.
- Transform between equivalent deterministic and non-deterministic finite automata, and regular expressions.
- Simplify automata and context-free grammars.
- Determine if a certain word belongs to a language;
- Define Turing machines performing simple tasks and classify problems as decidable or undecidable.

*Judgement and approach:*

- Differentiate and manipulate formal descriptions of languages, automata and grammars with focus on regular and context-free languages, finite automata and regular expressions.
- Differentiate among different models of computation based on their computation power.

**Syllabus:**

1. Regular languages: DFA, NFA, Subset construction, Regular, Pumping Lemma, DFA state minimization, Myhill-Nerode relations and theorem. (12 lectures)
2. Grammars: Production systems, Right linear grammar and Finite state automata, Context free grammars, Normal forms, Pumping Lemma for CFLs, Subfamilies of CFL, Derivation trees and ambiguity. (10 lectures)
3. Pushdown automata: Acceptance by final state and empty stack, Equivalence between push- down automata and context-free grammars, Closure properties of CFL, Deterministic push- down automata, the CKY algorithm. (10 lectures)
   Advanced topics: the Chomsky-Schützenberger theorem, Parikh’s theorem.
4. Turing machines: Techniques for Turing machine construction, Generalized and restricted versions equivalent to the basic model, Universal Turing machine, Recursively enumerable sets and recursive sets. (10 lectures)
5. Decidability: Decidable and undecidable problems, Reduction, Post's correspondence problem, Rice's theorem, decidability of membership, emptiness and equivalence problems of languages. (10 lectures)
   Advanced topics: Gödel’s incompleteness theorem, Gödel’s proof.

**Reference Books:**

1. D. C. Kozen, “Automata and Computability,” Springer, 1997.
2. J. E. Hopcroft, R. Motwani and J. D. Ullman, "Introduction to automata theory, languages and computation," Pearson, 3/e, 2006.
3. E. A. Rich, “Automata, Computability and Complexity: Theory and Applications,” Pearson,
   2007.
4. M. Sipser, “Introduction to the Theory of Computation,” Cengage Learning, 3/e, 2012.
5. Peter Linz, "An introduction to formal language and automata", 3rd edition, Narosa publishing house, 2002.