

**CS 4511: Computability and Complexity (4)****Catalog Description:**

Fundamentals of the mathematical theory of computation. Turing machines, Church-Turing Thesis, recursive and recursively enumerable languages, unsolvable problems, Rice's Theorem, deterministic and nondeterministic time and space complexity, complexity classes, NP-completeness, Cook's Theorem, P vs N.

**Textbook:** M. Sipser, *Introduction to the Theory of Computation*, 2<sup>nd</sup> Ed., Thompson Course Technology, 2006.

**Course Goals:**

This course introduces elements of the theory of computation, an active research area involving the formulation of precise questions and answers concerning what is computable, by what means, in what amount of space and time. Remarkably, such work does not depend essentially on any particular digital technology or programming language. Instead, computations are expressed and studied as mathematical objects. In this spirit, the course emphasizes standard methods for expressing and establishing mathematically precise claims. We introduce many well-known, widely-studied definitions and carefully consider what follows from them.

**Prerequisites by Course & Topic**

CS 3512 - predicates and quantifiers, sets, relations, proof techniques, recursion and induction, combinatorial counting techniques, strings, trees

**Major Topics Covered in the Course**

- Math Notation and Techniques
- Regular Languages and Finite Automata
- Context-Free Languages and Pushdown Automata
- Turing Machines, Recursive and RE Languages, Computability
- Decidability, reducibility
- Complexity, P, NP, NP-Completeness

**Class/Laboratory Schedule:** Lecture: 4 hours per week, Laboratory: 0

**Course Outcomes**

1. Further develop ability to understand, apply and prove precise mathematical claims using a range of fundamental definitions from discrete math and formal languages.
  - a. Understand and apply definitions and notation for sets, relations, functions.
  - b. Comprehend and analyze formal languages (sets of strings over finite alphabets), including Kleene closure.
  - c. Solidify ability to comprehend, synthesize and present proofs by mathematical induction.
  - d. Solidify comprehension of recursive definitions of sets, and learn to comprehend and synthesize proofs by structural induction (based on such definitions).
2. Detailed knowledge of foundational results for regular languages and finite automata.
  - a. Understand recursive definitions of regular languages regular expressions, and the use of regular expressions to represent regular languages.
  - b. Understand definitions and properties of finite automata, including the role of nondeterminism.
  - c. Detailed knowledge of the correspondence between regular languages and finite automata (Kleene's Theorem), including ability to carry out and reason about the various mathematical constructions used in proof of Kleene's Theorem.
  - d. Knowledge of characterization of minimal finite automata in terms of equivalence classes of strings, and associated algorithm for finding minimal FAs.
  - e. Pumping Lemma and other methods for proving that a language is not regular.
3. Knowledge of foundational results for context-free languages and pushdown automata..
  - a. Ability to represent context-free languages using context-free grammars, and ability to prove properties of context-free grammars.

- b. Knowledge of pushdown automata and the correspondence theorem for pushdown automata and context-free languages.
- c. Some familiarity with role of nondeterminism in context-free languages.
- d. Ability to apply the Pumping Lemma for context-free languages to show that a language is not context-free.
- 4. Turing machines and the concept of computability.
  - a. Ability to apply basic definitions related to Turing machines.
  - b. Familiarity with definitions of recursive and recursively enumerable languages.
  - d. Some understanding of the role of nondeterminism in Turing machines.
- 5. Decidable problems and reducibility.
  - a. Exposure to decision problems, including unsolvable problems such as the halting problem.
  - b. Reducibility and mapping reducibility.
- 6. Complexity
  - a. Time complexity and algorithms
  - b. The classes P and NP
  - c. NP-completeness
- 7. Ability to work individually on hard problems.

**Relationship to Program Outcomes**

CS 4511 is an elective course that may be taken upon successful completion of the introductory sequence and discrete math. This course contributes to meeting the following program outcomes:

1. *Students understand the mathematics and statistics that underlie scientific applications.*  
 Students further develop ability to understand, apply and prove precise mathematical claims using a range of fundamental definitions from discrete math and formal languages. They understand how these foundational mathematical techniques are used to develop the theory of computation. They are able to comprehend and synthesize proofs of precise claims about various aspects of the theory of computation, including proofs by mathematical and structural induction. Course outcomes 1, 5, 6 and 7 map to this program outcome.
3. *Students understand the fundamentals of computer organization and architecture, data structures and related algorithms, and programming languages*  
 Students learn the fundamentals of formal languages, including regular languages, finite automata, context-free languages and pushdown automata. This necessarily involves some consideration of the fundamental idea of an algorithm as it is made precise in the theory of computation. They also learn many algorithms (constructions) used to establish fundamental results in theory of computation. Course outcomes 2-6 map to this program outcome.
4. *Students can apply computer science principles and practices to a variety of problems.*  
 This course increases a student's knowledge of the theory of computation. Students develop their abilities to determine what is computable, by what means, and in what amount of space and time. They gain considerable experience in the precise statement and proof of claims about various forms of computation. Course outcomes 2-7 map to this program outcome.

**Assessment Plan for Course:**

This course is assessed every third year by the instructor and a course assessment document covering all of the course outcomes and their effect on the program outcomes is prepared.

**Estimate CSAB Category Content**

	CORE	ADVANCED		CORE	ADVANCED
Data Structures			Computer Organization and Architecture		
Algorithms		1	Concept of Programming Languages		2
Software Design					

**Coordinator/Prepared by:** D. Dunham