

CS 3512: Computer Science Theory(4)**Catalog Course Description**

Sets, relations, functions. Recursive definitions of functions and sets. Proof methods, including mathematical and structural induction, diagonalization. Program correctness, asymptotic time/space complexity. Formal language theory, including regular languages and expressions, deterministic/nondeterministic finite automata, Kleene's Theorem..

Textbook: J. Hein, *Discrete Structures, Logic, and Computability*, 3rd edition, Jones and Bartlett, 2009.

Course Goals:

The objective of this course is to introduce students to standard elements of discrete mathematics and formal language theory relevant to computer science, and to develop a firm foundation for further study, either as students or as professionals.

Prerequisites by course and topic:

Math 1296 or 1596 – basic mathematical reasoning

CS 2511 – significant experience with design and implementation of algorithms with high-level programming languages

Major Topics Covered in the Course

- Sets
- Relations
- Functions
- Recursive definitions
- Proof methods
- Program correctness
- Complexity
- Formal language theory
- Regular languages and expressions
- DFAs/NDFAs
- Kleene's Theorem

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

Course Outcomes

1. Further develop the ability to use standard mathematical definitions from discrete math and formal language theory.
 - a. Set theory and notation, including infinite union and intersection.
 - b. Tuples, Cartesian products, relations, lists, binary trees.
 - c. Functions, including image, pre-image, injection, surjection, bijection, inverse, composition.
 - d. Strings, languages, language products, Kleene closure.
2. Passing familiarity with countability and diagonalization.
3. Experience with standard uses of recursion and induction.
 - a. Inductive definitions of sets.
 - b. Recursive definitions of functions.
 - c. Structural induction.
 - d. Inductive proofs of correctness of recursive functions.
 - e. Mathematical induction, strong and weak.
4. Mathematical understanding of asymptotic notation: big-O, big-Omega, big-Theta.
5. Exposure to formal language theory.
 - a. Regular languages and regular expressions.
 - b. Deterministic and nondeterministic finite automata.
 - c. Kleene's Theorem.
 - d. Brief discussion of computability and complexity.
6. Some experience with counting, mostly as a means of checking understanding of definitions.

7. Strengthen ability to read and understand precise mathematical claims and their proofs. This is a main emphasis throughout the course, intended to prepare students for lifelong appreciation of mathematical material related to computer science.

Relationship to Program Outcomes

This course contributes to meeting the following program outcomes:

1. *Students understand the mathematics and statistics that underlie scientific applications.*

Students further develop their ability to understand the mathematics underlying theoretical computer science. In particular, students increase their ability to understand mathematically precise claims related to computer science, and improve their ability to understand and write proofs of such claims. We can't hope to teach all the math they might need, but we can help prepare them to learn more on their own. All course outcomes map to this program outcome.

3. *Students understand data structures and related algorithms, programming languages, and the fundamentals of computer organization and architecture.*

Students are exposed to some of the fundamental elements of formal language theory, and are well prepared for further study in this area. Students gain some experience with mathematically adequate accounts of lists and binary trees, including experience with relevant proof techniques. Students will also hear about the theoretical limitations of algorithm design such as the halting problem and the Church-Turing Thesis. All course outcomes map to this program outcome.

4. *Students can apply computer science principles and practices to a variety of problems.*

Students learn to construct proofs of correctness of recursive programs and learn the mathematical basis asymptotic notation, which prepares them to understand claims concerning the time and space requirements of algorithms. Course outcomes 2, 3, 4 and 6 map to this program outcome.

6. *Students can communicate effectively both orally and in writing.*

Students gain significant experience writing standard mathematical English, and in presenting clearly written arguments about mathematically precise concepts. All of the course outcomes contribute 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	1		Computer Organization and Architecture		
Algorithms	1		Concept of Programming Languages	1	
Software Design					

Coordinator/Prepared By: H. Turner