

# CS 536: Science of Programming

## Course Description

We will study how to formally specify how programs execute (operational semantics), how to describe what mathematical functions programs compute (denotational semantics), and how to use logic to characterize properties and invariants of the program execution (axiomatic semantics). Using these specifications, we can prove interesting and relevant properties of programming languages: for example, that a programming language has a sound type system, or that a compiler preserves type safety. These techniques have significant practical utility and are becoming increasingly relevant to other areas of computer science.

## Objectives

- The goal of this course is to study formal techniques for describing computation and compilation.
- This approach leads to a more general understanding of programming languages, specification, logic, mathematics, and proof theory.
- We apply formal reasoning to nondeterministic programs and to concurrent programs, and provide an introduction to reasoning about distributed systems (temporal logic).

## Prerequisites

- CS 331 or CS 401 or CS 403.

## Syllabus

- An overview of propositional logic and predicate calculus
- Semantics of programming languages
- Weakest precondition semantics for sequential programming constructs
- Weakest precondition semantics for non deterministic programming constructs
- Formal verification of sequential and non deterministic programs
- The process of developing programs from their proofs
- A Hoare Logic for shared-variable concurrency
- A Hoare Logic for synchronous message passing
- Transformational design and Hoare logic

Edited March 2006 ([html](#), [css](#) checks)