

EECS 396: Formal Verification of Software Systems

Introduction

This course introduces students to the concepts and tools necessary for the construction of high-reliability software systems through formal methods. Concepts presented in this course include the role of semantics, major specification and verification approaches, and feasible automated and semi-automated support. The study of theoretical verification techniques is supplemented with hands-on practice in writing and checking precise software specifications using practical tools supporting these techniques. The course will be given in lecture form, two per week for approximately 14 weeks. The course will use material from the following references:

- D. Peled, *Software Reliability Methods*, Springer Verlag, 2001.
- M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University press, 2000.

Prerequisite

The course assumes familiarity with basic discrete mathematics and computer science concepts including relations, functions, graph theory, propositional and predicate logic, and finite-state machines.

Course syllabus

1 Introduction

2 Modeling Software Systems

- Sequential, Concurrent and Reactive Systems.
- Floyd-Hoare Logic: Axioms and rules; Soundness and completeness.

3 Deductive Software Verification

- Axiomatic Program Verification.
- Partial and Total Correctness. Proof rules for partial correctness.
- Verification of Concurrent Programs. Compositionality.
- Soundness and Completeness of Proof Systems
- Theorem Proving tools: PVS

4 Process Algebra and Equivalences

- Process Algebras. A Calculus of Communicating Systems.
- Equivalences between Agents: Trace equivalence, Failure equivalence, and Simulation Equivalence.
- Bisimulation and Weak Bisimulation equivalence. Calculating Bisimulation Equivalence.
- Studying Concurrency using Process Algebra.
- Process Algebra Tools: LOTOS.

5 Combining Testing and Formal Methods

- Abstraction Techniques
- Combining Testing and Model Checking
 - Direct Checking
 - Black Box Checking: Deadlock Detection
- Predicate based testing

Grades

To encourage hands-on experience, there will be several assignments involving the use of software verification tools. In addition, each student will have to complete a research project for a complete specification and verification of a medium size software system. Reports on the projects will be written up, and results will be presented in class.

- Assignments 60%
- Project 30%
- Project presentation 10%

Credit hours

Students will receive 3 hours of credit for this course.