COMS 6998-3, Fall 2009: Formal Verification of Hardware and Software Systems

Instructors: Michael Theobald (michael.theobald@gmail.com) and Franjo Ivančić (ivancic@gmail.com)
Class: Mondays, 6:10 PM – 8 PM
Office hours by appointment
Course Homepage: available at courseworks.columbia.edu

> **Note: If you are interested in taking the course, then the instructors would love to hear from you! Please send a brief note by email. The course was added late and the instructors would like to get an idea of the size of the class.**

Motivation

The evolution of computing systems is witnessing two trends: Computing systems are becoming ever more complex, while also becoming ever more ubiquitous as the increasing omnipresence of embedded systems in today's society shows. However, the design and implementation of ever more complex systems with a high degree of confidence in the correctness of the system in all scenarios is hindered by the ability of designers to reason about these complex systems. In many cases, new designs are still analyzed only using simulation or testing of selected test cases or scenarios. While these simulation-based techniques are often effective in the early design stages when designs still have many bugs, the efficacy drops alarmingly as designs become cleaner. Subtle bugs, such as bugs that depend on particular schedules in a concurrent software program, are often missed due to the limitations faced by testing-based approaches.

Course Description

The course introduces the theory and practice of formal methods for the design and analysis of concurrent and embedded systems. Model checking and related computer-aided verification techniques are emerging as practical alternatives to simulation for debugging complex systems. These methods allow the designer to verify that a mathematical model of a system satisfies its abstract logical specification. This approach has been most effective in the analysis of control-intensive hardware components, and is rapidly becoming an integral part of the design cycle. More recently, similar techniques have been developed for the analysis of software and have been applied to real-world programs such as device drivers.
The topics of the course will include temporal logics, deadlock/liveness, fairness, model checking, SAT checkers, BDDs, symbolic model checking, abstraction techniques, an introduction to the formal verification tools SPIN (Protocols/Software) and SMV (Hardware). We also plan to cover case studies/experiences on how formal verification is currently gaining momentum in industry. Additional topics may include an introduction to assertion-based verification, methodology aspects, and techniques that combine formal verification and simulation.

Prerequisites

The course targets CS as well as Computer Engineering and EE students. Ph.D. students, M.S. students as well as advanced undergrads are all welcome. Familiarity with basic propositional logic, finite automata/state machines, and search algorithms (depth and breadth first search) would be useful. These concepts are usually taught in any data structures, algorithms or discrete mathematics course. If you are unsure whether you have sufficient background for the course, please contact the instructors.

Reference Books

There is no required textbook for this course. However, the following books are recommended:

- Authors: Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled
  Title: Model Checking
  Publisher: MIT Press
  Year: January 2000

- Author: Gerard Holzmann
  Title: The SPIN Model Checker: Primer and Reference Manual
  Publisher: Addison-Wesley
  Year: September 2003

Grading

There will be optional homework assignments. The major component of the evaluation besides class participation will be the final class project. Each student may select a project that is adequate for his/her background. The class project requires a presentation in the final week of classes.
Projects can be of various forms:

- Research Project: Present a research paper from a recent Formal Verification Conference.
- HW Application: Apply Formal Verification to a real hardware design.
- SW Application: Apply Formal Verication to a program.
- Programming: Implement a fundamental Formal Verification algorithm.

About the Instructors

Dr. Michael Theobald is currently with D. E. Shaw Research. He works on the formal verification of a new specialized supercomputer. Michael was a Postdoctoral Fellow in the Model Checking Group at Carnegie Mellon University from 2002-2004. He received his Ph.D. from Columbia in 2002.

Dr. Franjo Ivancic is a Research Staff Member at NEC Laboratories America in Princeton, NJ. In 2003, he received his Ph.D. from the University of Pennsylvania. His research interests include software verification, model checking, hybrid systems, and the design automation for embedded software.