Instructor: Swarat Chaudhuri
Time: TR 4:00pm - 5:15pm
Location: 1042 Duncan Hall
Office hours: 3103 Duncan Hall; TR 2:30-3:30pm

http://www.cs.rice.edu/~swarat/COMP507

Software is getting more and more complex by the day; at the same time, our reliance on software in all areas of our lives is increasing. In this world, it is more than important to make software more reliable. The field of computer-aided program verification takes on this challenge, developing algorithmic methods that can give guarantees about the dynamic behavior of programs by examining the programs’ code.

This course will attempt a broad but hands-on study of program verification. Topics will include the main approaches to program verification, and will cover theoretical foundations as well as practical tools.

Assignments and evaluation

Your grade in this course will be entirely based on 6-7 homework assignments. Each of these assignments will involve programming a verifier of some sort. Some of them will additionally contain some theoretical exercises.

The programming assignments are be to done in groups of 2. During each assignment, we will (semi)-randomly assign you a coding partner. If the total number of students taking the course is odd, one group (randomly selected) will have 3 students.

Programming assignments will be graded via an in-person codewalk, where every member of the coding team must be present. You must be prepared to answer questions related to design and implementation choices.

The written parts of the assignments are to be completed and submitted individually. Discussions about these problems are permitted but not encouraged.
Programming language and textbook

All programming in course will be done in the Ocaml language, and using the CIL infrastructure for C program analysis:

http://caml.inria.fr/
http://www.cs.berkeley.edu/~necula/cil/

Here is a book on Ocaml:

http://caml.inria.fr/pub/docs/oreilly-book/

We won't follow any one textbook in this course; however, we will often refer to the following text:

- **The Calculus of Computation.** By Aaron Bradley and Zohar Manna. Springer 2007. (The book is available online through the library website.)

In addition, we will use papers and surveys available online.

List of topics (tentative)

**Unit 1: Types and dataflow analysis**

- Introduction to functional programming in Ocaml. (especially higher-order features and closures)
- The simply typed lambda calculus. Writing a type checker.
- Hindley-Milner type inference. Type inference as a fixpoint computation.
- Types for imperative languages: Information flow type system in JFlow/Jif. Introduction to CIL.
- Dataflow analysis. More on CIL.
- Interprocedural dataflow analysis

**Unit 2: Model checking**

- The special case of finite-state programs and model checking. Temporal logics and omega-automata.
- Automata-theoretic model checking using SPIN
- Mu-calculus and symbolic model checking.
- SAT-based bounded model checking.

**Unit 3: Proving programs correct**
• Recapitulation of first-order logic and first-order theories. Decision procedures for first-order theories: uninterpreted functions, linear arithmetic.
• The Yices SMT-solver
• Hoare logic and partial correctness
• Termination and total correctness
• Implementing a program verifier
• Data structures and separation logic
• Concurrency and Owicki-Gries logic
• Review and in-class proofathon

Unit 4: Abstract interpretation and invariant generation

• Abstract domains. Interval analysis
• Abstract interpretation using polyhedra and octagons
• Galois connections and theory of abstract interpretation
• Predicate abstraction Part 1: SLAM
• Predicate abstraction Part 2: Liquid types
• Invariant generation using interpolants
• Termination analysis: Terminator

Unit 5: Counterexample generation

• From symbolic execution to test generation: DART
• Enumerative bug-finding: Chess