Virginia Tech

Virginia Tech ❖ Bradley Department of Electrical and Computer Engineering

ECE 4984 and 5984: Certified Programming

Course Syllabus **Fall 2017**

Instructor

TBA.

1 Overview

We say that a piece of software is certified when its behavior has been mathematically modeled and proven to comply with a specification of the software's requirements. With software defects costing the US economy billions of dollars annually, producing certified software is today of high importance.

Writing pen and paper proofs about software is too tedious and error-prone (it's easy to miss a case) to be practical, and fully automated analysis tools are not generally applicable due to the undecidability of most software analysis tasks. To achieve trustworthiness and wide applicability, one can develop certified programs using an interactive theorem prover (ITP). The Isabelle/HOL ITP combines automatic proof methods with interactive construction of proofs by the user, as well as automatic code generation from specifications, rapid prototyping tools, and an advanced IDE.

While developing software using an ITP is an expensive process, the situation is rapidly improving, as exemplified in recent breakthroughs in ITP technology and applications, such as the certification of the seL4 operating system, the CompCert compiler, and the IronFleet suite of distributed systems.

The objective of this course is to teach students the basic techniques for developing certified software with an ITP like Isabelle/HOL. After an initial tutorial on Isabelle/HOL, the course will cover top-down certified development of software from specifications to efficient functional programs, with application to building certified distributed systems, and bottom up proof of systems software written in the C language. The course will also cover the use of lightweight prototyping and testing tools that speed up the certified development process. Students will apply their knowledge to new application domains by completing a course project over a period of 8 weeks.

Upon completion of the course, the student will be able to:

- Use Isabelle/HOL as a functional-programming environment: write high-level specification and refine them to efficient purely functional programs.
- Use Isabelle/HOL's Nitpick tool, Quickcheck tools, and code generation to rapidly prototype and check the correctness of software designs, functional programs, distributed system designs, and C code.
- Develop certified distributed systems in Isabelle/HOL and deploy them.
- Develop certified systems software written in the C programming language.

2 Prerequisites

- 5984: Graduate standing.
- 5984 and 4984: Good programming skills in at least one programming language. Basic notions in set theory and propositional logic, and finite automata theory.

It is preferable but not mandatory to know the basics of some functional programming language (e.g. OCaml, Haskell, Lisp, Scala, F#, etc.), roughly corresponding to the material covered in chapters 1 to 6 of the book Introduction to Objective Caml.

3 Course meeting time and location

TBA.

4 Required and Recommended Texts

The only required book is Concrete Semantics with Isabelle/HOL (freely available), by Tobias Nipkow and Gerwin Klein. The following two books are recommended but not necessary to follow and complete the course:

- John Harrison. Handbook of practical logic and automated reasoning.
- P.B. Andrews. An Introduction to Mathematical Logic and Type Theory.

5 Grading (tentative, to be confirmed)

The grade will be split in 50% for programming and proving exercises and 50% for a course project.

The course project will allow students to put their knowledge to practice by developing certified software for an application of their choice. This may consist in using the techniques seen in the class to certify a more complex piece of software (undergraduate projects), or in developing new infrastructure for producing certified software in a particular application domain and demonstrating the infrastructure on a realistic example (graduate projects).

There will be three kinds of projects to choose from: custom projects, graduate projects, and undergraduate projects. Custom projects are proposed by the students and refined with the help of the instructor, who must approve them before students are allowed to use them as course project. Graduate and undergraduate projects are proposed by the instructor. Graduate projects will require independent study of research material, whereas undergraduate projects will come with a predetermined set of steps to accomplish to complete the project. Undergraduate students can choose to complete any of the three types of projects, while graduate students can only choose a custom or graduate project, and a graduate student's custom project must include independent study of research material.

Students will have 8 weeks to complete the course project. Programming and proving exercises will be handed out weekly, and must be completed by the time indicated on each handout.

Course Topic Outline (tentative, to be confirmed)

Topic	"Concrete Semantics" Chapters
Basic programming and proving in Isabelle/HOL: functional programming, inductive definitions, induction proofs, simplification, automatic proof methods, deduction in Isabelle/Pure, the Isar proof language, locales.	Part 1.
Semantics of imperative programs and Hoare Logic; application to proving C code correct.	Chapters 7 and 12.
Proving C programs with Autocorres.	Not covered
Code generation from Isabelle/HOL theories through higher-order rewriting.	Not covered
Type definitions, lifting and transfer, code generation with data-type refinement.	Not covered
Prototyping specifications and programs using Nitpick and Quickcheck.	Not covered
Modeling and developing certified distributed systems in Isabelle/HOL; I/O-automata, ghost variables and simulation proofs.	Not covered