The University of Iowa Fall 2015

CS:5810

Formal Methods in Software Engineering

Course Overview

Staff

- Instructor: Prof. Cesare Tinelli

- TA: Ruoyu Zhang

Course Info and Material

- All information, including the syllabus, available at:

  `http://www.cs.uiowa.edu/~tinelli/5810`

- No required textbook
- Class notes and additional reading material to be posted on the website
- Announcements and discussions on Piazza
- Check the course website and the Piazza website regularly!

Course Design Goals

1. Understand how formal methods (FM) help produce high-quality software
2. Learn about formal modeling and specification languages
3. Write and understand formal requirement specifications
4. Learn about main approaches in formal software verification
5. Know which formal methods to use and when
6. Use automated and interactive tools to validate models and code

Course Topics

Software Specification

- High-level semantic design
- System design and behavioral properties
- Code-level properties

Main Software Validation Techniques

- Model Finding/Checking:
  often automatic, abstract
- Deductive Verification:
  typically semi-automatic, precise (source code level)
- Abstract Interpretation:
  automatic, correct, not complete, terminates

Course Organization

- The course is organized by level of specification
- Emphasis on tool-based specification and validation methods
- A number of ungraded exercises
- Hands-on homework where you specify, design, and verify
- For each main topic
  - A team introductory homework asignment
  - A team mini-project
- 1 midterm, 1 final exam
- More details on the syllabus and the website

Part I: High-level Design

Language: Alloy

- Lightweight modeling language for software design
- Amenable to a fully automatic analysis
- Aimed at expressing complex structural constraints and behavior in a software system
- Intuitive structural modeling tool based on first-order logic
- Automatic analyzer based on SAT solving technology

Learning Outcomes

- Design and model software systems in the Alloy language
- Check models and their properties with the Alloy Analyzer
- Understand what can and cannot be expressed in Alloy

Part II: Model-based Development

Language: Lustre

- Executable specification language for synchronous reactive systems
- Designed for efficient compilation and formal verification
- Used in safety-critical applications industry
- Automatic analysis with tools based on model-checking techniques

Learning Outcomes:

- Write system and property specifications in Lustre
- Perform simulations and verifications of Lustre models
- Understand what can and cannot be expressed in Lustre

Part III: Code-level Specification

Languages: Dafny, ACSL

- Programming languages with specification constructs
- Specifications embedded in source code as formal contracts
- Tool support with sophisticated verification engines
- Automatic analysis based on theorem proving techniques

Learning Outcomes:

- Write formal specifications and contracts in such languages
- Verify functional properties of programs with automated tools
- Understand what can and cannot be expressed in these languages