The University of Iowa

22c181: Formal Methods in Software Engineering

Spring 2011

Course Overview

Copyright 2010-11 Cesare Tinelli. These notes are copyrighted materials and may not be used in other course settings outside of the University of Iowa in their current form or modified form without the express written permission of one of the copyright holders.
Staff

• Instructor: Cesare Tinelli
  • Office hours (201D MLH):
    Mon 3:30–5pm, Thu 4:00–5:30pm, and by appointment

• Teaching Assistant: Ben Berman
  • Office hours (201C MLH):
    Tue, Thu 10:30-12noon, and by appointment
Course Info and Material

• All the relevant information about the course, including the syllabus, will be available on this website:
  
  http://www.cs.uiowa.edu/~tinelli/181/
  
• Textbook: none.

• Class notes and additional reading material will be posted on the website


• Distance education students will also have access to recorded lectures

• Check the website regularly!
Course Design Goals

- Understand how formal methods (FM) help produce high-quality software
- Learn about formal modeling and specification languages
- Write and understand formal requirement specifications
- Learn about main approaches in formal software verification
- Know which formal methods to use and when
- 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

Software Validation

• Model Finding/Checking: often automatic, abstract
• Deductive Verification: typically semi-automatic, precise (source code level)
Course Organization

• Most of the course devoted to high-level semantic design and code-level properties
• Emphasis on tool-based specification and validation methods
• A few ungraded exercises
• Hands-on homeworks where you specify, design, and verify
• For each main topic
  • An individual introductory homework
  • A team mini-project
• 1 written 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 what 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 what cannot be expressed in Lustre
Part III: Code-level Specification

Languages: JML and/or Dafny

- Behavioral interface specification for Java/Dafny modules
- Based on the design by contract paradigm
- Formal semantics, same syntax as host programming lang.
- Specifications embedded in source code
- Various verification tools available

Learning Outcomes:

- Write formal specifications and contracts in JML/Dafny
- Understand how source code and specs are represented in logic
- Verify functional properties of programs with automated tools