22c181: Formal Methods in Software Engineering
The University of Iowa
Spring 2010

Course Overview
Staff

- Instructor: Cesare Tinelli
  - Office hours (201DMLH):
    Mon 11-12:30pm, Thu 4:00-5:30pm, and by appointment

- Teaching Assistant: Jeff Story
  - Office hours (B20J MLH):
    Wed 2:30-4pm and Fri 11-12:30pm
Course Info and Material

- All the relevant information about the course, including the syllabus, will be available on this website:
  


- The required textbook is mostly for background knowledge

- 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
- Several 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: Model Design and Checking with 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 II: High-level Design with Alloy

- Lightweight modelling 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 Alloy models and their properties with the Alloy Analizer
- Understand what can and what cannot be expressed in Alloy
Part II: Code-level Specification with JML

- Behavioral interface specification language for Java modules
- Based on the design by contract paradigm
- Rigorous formal semantics, Java syntax
- Specifications embedded as annotation comments in Java programs
- Various verification tools available

Learning Outcomes:

- Write formal specifications and contracts in JML
- Understand how Java and JML can be represented in logic
- Verify functional properties of Java programs with JML tools