22c181: Formal Methods in Software Engineering

Spring 2008

Course Overview

The University of Iowa
Staff

- Instructor: Cesare Tinelli
  - Office hours: Tue 4:00-5:30pm, Fri 2:00-3:30pm, and by appointment.

- Teaching Assistant: George Hagen
  - Office hours (in the lab): Wed 2:30-3:30pm, Thu 1:30-2:30
Course Info and Material

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

- There is no textbook for this course

- Class notes and related reading material will be posted on the website

- Long distance education students will also have access to recorded lectures

- Check the website at least every other day!
Course Design Goals

- Understand how formal methods (FM) help to produce high-quality software

- Understand the difference between:
  - automatic vs interactive formal verification
  - concrete vs abstract system models

Illustrate main approaches in formal software verification today

- Know when and which formal methods to use

- Write and understand formal requirement specifications

- Use automated and interactive tools to produce formal proofs

- Avoid overburdening with formal details — Yet enough formality to let participants know what they are doing
Major paradigms for formal validation of software:

- **Model Checking** (here using temporal induction)
  automatic, abstract, not so expressiveness

- **Deductive Verification:**
  semi-automatic, precise (source code level), expressive

- **Automatic Test Case Generation:**
  complements model checking and deductive verification
Course Organization

Organization

- Most of the course devoted to first two MC and DV
- Will do ATCG time permits
- Hands-on lab assignments where you specify, design, and verify
- Several ungraded exercises
- 2-3 graded mini-projects for teams of 2
- 1 written midterm, 1 final exam
- More details on the syllabus and the website
Part I: Model Checking with Lustre

- Synchronous, declarative real-time programming language
- Designed for efficient compilation and formal verification
- Used in safety-critical applications industry:
  - Aerospatiale: Airbus A310–340
  - Eurocopter: World-leading civil helicopter manufacturer
  - Schneider Electric: Nuclear power plant control
  - Rockwell-Collins: Major avionics company

Learning Outcomes:

- Write formal system and property specifications in Lustre
- Execute simulation and verification of Lustre models
- Understand what can and what cannot be expressed in Lustre
Part II: Deductive Verification with KeY

- Integrated UML-based CASE tool/verification system: conventional and formal development of OO software
- Frontend: commercial CASE tool Borland Together
- Specifications written in Object Constraint Language (OCL)
- Verification of sequential Java programs (no floats)
- Background knowledge: Java, UML basics (class diagrams)

Learning Outcomes:
- Write formal specifications and contracts in OCL
- Understand how Java and OCL can be represented in logic
- Verify functional properties of Java programs with KeY