

The University of Iowa

Fall 2021

**CS:5810**

**Formal Methods in Software Engineering**

## **Course Overview**

Copyright 2021, 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



## COURSE INFO AND MATERIAL

---

- All information, including the syllabus, available at:  
<http://www.cs.uiowa.edu/~tinelli/classes/5810/Fall121>
- Textbook (draft): *Program Proofs* by Rustan Leino, 2020
- Class notes and additional reading material to be posted on the website
- Recorded lectures on UICapture
- Announcements and discussions on Piazza
- Submissions and grades on ICON
- Check the course website and the Piazza website **regularly!**

## COURSE DESIGN GOALS

---

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

## COURSE TOPICS

---

### Software Specification

- High-level design
- System-level design (Model-based Development)
- Code-level design

### Main Software Validation Techniques

- **Model Checking:**  
often automatic, abstract
- **Deductive Verification:**  
typically semi-automatic, precise (source code level)
- **Abstract Interpretation:**  
automatic, correct, incomplete, terminating

## COURSE ORGANIZATION

---

- Course organized by level of specification
- Emphasis on tool-based specification and validation methods
- A number of graded and ungraded exercises, in class and at home
- Hands-on homework where you specify, design, and verify
- For each main topic
  - An introductory homework assignment
  - 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 relational logic
- Automatic analyzer based on SAT solving technology

## 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 relational 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

## 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

---

### Language: Dafny

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

## PART III: CODE-LEVEL SPECIFICATION

---

### Language: Dafny

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

### Learning Outcomes:

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