EECS498-008

Formal Verification of Systems Software

Instructor
Manos Kapritsos
https://web.eecs.umich.edu/~manosk/

Class meeting times (in person)
MW 3-4:30pm (Lecture)
F 10:30-11:30 (Discussion)

Course description
For the past 60 years, we have been relying on testing for building robust, bug-free software. And yet, despite our best efforts, bugs slip past our test cases all the time, resulting in a number of incidents that range from benign to catastrophic. In this class, you will learn an entirely different way to build robust software: by writing code that is formally proven to be **free of bugs**. We will approach this problem from a systems perspective, focusing on how to design complex (often distributed) systems, while still being able to reason about their correctness. At the same time, the class will give you hands on experience with Dafny, one of the most promising tools for practical formal verification of systems software.

During this course, you will learn how to formally *specify* a system's behavior, how to *prove* that the high-level design of the system meets that specification and finally how to show that the system's low-level implementation retains those properties. The course does not assume any prior knowledge in formal verification. We will start from the basics of the Dafny language and build from there. In the end, you should be able to design and prove correct a complex system.

Learning objectives
By the end of the course, you should:

- Understand the fundamentals of formal verification (specification, proof, invariants, etc.)
- Be familiar with Dafny at both the protocol and implementation level
- Be able to design a new system as a state machine in Dafny
- Understand the concept of an inductive invariant and how to find one
- Be able to prove the correctness of a protocol-level state machine
- Understand the concept of refinement
- Be able to perform a refinement-based proof on a complex system
- Be familiar with more advanced topics such as asynchronous APIs and proof automation

Tentative list of assignments

- Exercise set #1 (done individually)
    - Dafny basics: datatypes, assertions, predicates, etc.

    - Recursion
    - Loop invariants
    - Specification
- Exercise set #2 (done individually)
    - State machines
    - Inductive invariants
    - Distributed systems
- Project 1 (groups of 2)
    - Design and prove correct a Distributed Lock service
- Exercise set #3 (done individually)
    - Refinement
- Exercise set #4 (done individually)
    - Asynchronous API
    - Application correspondence
- Project 2 (groups of 2)
    - Design a Sharded Hash Table (SHT) protocol and prove it is correct using refinement

Grading policy (subject to change)

- Exercise sets: 30%
    - Set #1: 8%, Set #2: 8%, Set #3: 8%, Set #4: 6%
- Projects: 30%
    - Project 1: 15%
    - Project 2: 15%
- Exams: 40%
    - Midterm exam: 20%
    - Final exam: 20%

Tentative schedule

Week of August 29

Lecture 1: Intro Lecture 2: Dafny mechanics (datatypes, predicates, assertions)

Week of September 5

Labor Day Lecture 3: Dafny mechanics (recursion, loop invariants)

Week of September 12

Lecture 4: Specification Lecture 5: Specification and state machines

(Exercise set #1)

Week of September 19

Lecture 6: State machines and behaviors Lecture 7: Proving properties and inductive invariants

Week of September 26
Lecture 8: Leader election demo Lecture 9: Modeling distributed and async systems
(Exercise set #2)

Week of October 3
Lecture 10: Modeling distributed and async systems Lecture 11: Recap of first part
(Project 1)

Week of October 10
Lecture 12: Project 1 Lecture 13: Midterm exam

Week of October 17
Fall break Lecture 14: Refinement

Week of October 24
Lecture 15: Project 1 review Lecture 16: Asynchronous specs
(Exercise set #3)

Week of October 31
Lecture 17: Asynchronous specs (cont.) Lecture 18: Application correspondence
(Exercise set #4)

Week of November 7
Lecture 19: Multi-level refinement Lecture 20: Modules and automation

Week of November 14
Lecture 21: Cross-host concurrency Lecture 22: The future of systems verification
(Project 2)

Week of November 21
Lecture 23: TBD Thanksgiving break

Week of November 28
Lecture 24: TBD Lecture 25: Case study: IronFleet

Week of December 5
Lecture 26: Case study: VeriBetterKV Lecture 27: Recap of second part