

CS 472 — Provably Correct Programming

William Mansky

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Welcome to the Course!

- This is CS 472, Provably Correct Programming
- I'm glad you're here!
- Meets TR 11:00 AM – 12:15 PM in CDRLC 2406
- My office hours: M 10-11 AM, R 2-3 PM, and by appointment, in CDRLC 4450 and on Zoom via Blackboard
- TA office hours: TR 3-4 PM, location TBA and on Zoom
 - Office hours are great for homework help, or just to say hi!

Course Information

- Professor: William Mansky (he/him) (mansky1@uic.edu)
- TA: Eli Whitehouse (ewhit37@uic.edu)
- Prerequisites: CS 341 (functional programming)
- Website: <https://www.cs.uic.edu/~mansky/teaching/cs472/sp26/>
- Anonymous in-class questions: <https://pollev.com/wmansky771>
- In-person lectures
- Discussion board on [Piazza](#), assignments via [Gradescope](#) (entry code X8GY8B)

Asking questions

- In class: raise your hand anytime
- You can ask questions anonymously with PollEverywhere (<https://pollev.com/wmansky771>)
- On [Piazza](#)
 - Can ask/answer anonymously
 - Can post privately to instructors
 - Can answer other students' questions
- In office hours
- If you have a question, someone else probably has the same question!

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Software works badly

United Airlines says the outage that held up departing flights was not a cybersecurity issue



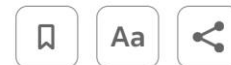
...to grow large enough to overwrite data that could cause unintended acceleration
[wikipedia.org/wiki/2009%E2%80%932011_Toyota_vehicle_recalls](https://en.wikipedia.org/wiki/2009%E2%80%932011_Toyota_vehicle_recalls)

NASA engineers finally came to the conclusion that there were too many files on the hard drive
[https://en.wikipedia.org/wiki/Spirit_\(rover\)](https://en.wikipedia.org/wiki/Spirit_(rover))

Explainer: What happened to shut down Toyota's production in Japan?

Reuters

August 30, 2023 6:27 AM CDT · Updated 4 months ago



Knight Capital Says Trading Glitch Cost It \$440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356

Runaway Trades Spread Turmoil Across Wall St.



[https://www.bloomberg.com/news/articles/2012-08-02/knight-capital-says-trading-glitch-cost-it-440-million](#)



[https://www.bloomberg.com/news/articles/2012-08-02/knight-capital-says-trading-glitch-cost-it-440-million](#)

What goes wrong?

- All these people tested their code!
- Corner cases
- Unexpected interactions
- Language gotchas
- Classic bugs: buffer overflows, use after free, ...
- Heisenbugs (test mostly succeeds, sometimes randomly fails)

Why can't we find bugs automatically?

```
for(i = 0; i < bound; i++){  
    a[i] = get_val(i);  
}
```

- To prove it doesn't overflow: show **bound** is $\leq$ length of **a**

Why can't we find bugs automatically?

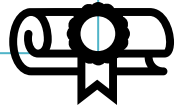
```
for(i = 0; i < bound; i++){  
    a[get_index(i)] = get_val(i);  
}
```

- To prove it doesn't overflow: need to prove **get_index** always returns reasonable values!
- *Safety* is entangled with *functional correctness*

Can we guarantee a program bug-free?

- Prove programs correct!

```
i = 1;
while(i <= n) {
    r = r * i;
    i++;
}
```



This course:

How can we write these proofs?

How can we write programs so they're easier to prove?

- This program will:
 - never overflow its stack
 - never dereference a null pointer
 - never call a function without meeting its preconditions
 - always return the right result!

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Grading

- In-class exercises: 25%
- Assignments: 50%
- Project: 25%

Exercises

- In each class, we'll work through some example problems/proofs
- Submit via [Gradescope](#)
- Due at the start of the next class
- You get credit as long as you make some progress on the problem
- Feel free to discuss with your neighbors, ask questions, suggest other approaches, etc.!

Assignments

- Programming/proving assignments
- Submit via [Gradescope](#)
- Due at 11:59 PM on the due date
- You can discuss strategy with other students, but don't look at each other's code!
- Always cite your sources (websites, other students, StackOverflow, ChatGPT, etc.)
- You'll get most of the credit for attempting a problem, even if you don't finish it – do what you can, and we'll work through tricky ones in class after the deadline

Online Sources and ChatGPT

- You may be able to find solutions online/generate them, but:
 1. It'll make the class harder; the assignments are there to help you learn.
 2. Always cite your sources! (websites, GPT, etc.)
 3. Please don't make your solutions public (e.g. on GitHub)
 4. There's less theorem-prover code out there than other kinds of code, so what you find will often be outdated or unreliable
 - a. ChatGPT is a *plagiarism machine*, so google your code!
 5. Always *run your proofs* and see if they work! (and if they don't, don't submit them!)

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Agenda

1. How do we prove things?
2. How do we prove things *about programs*?
3. What kinds of programs are interesting/tricky to prove?
4. Let's prove some programs!

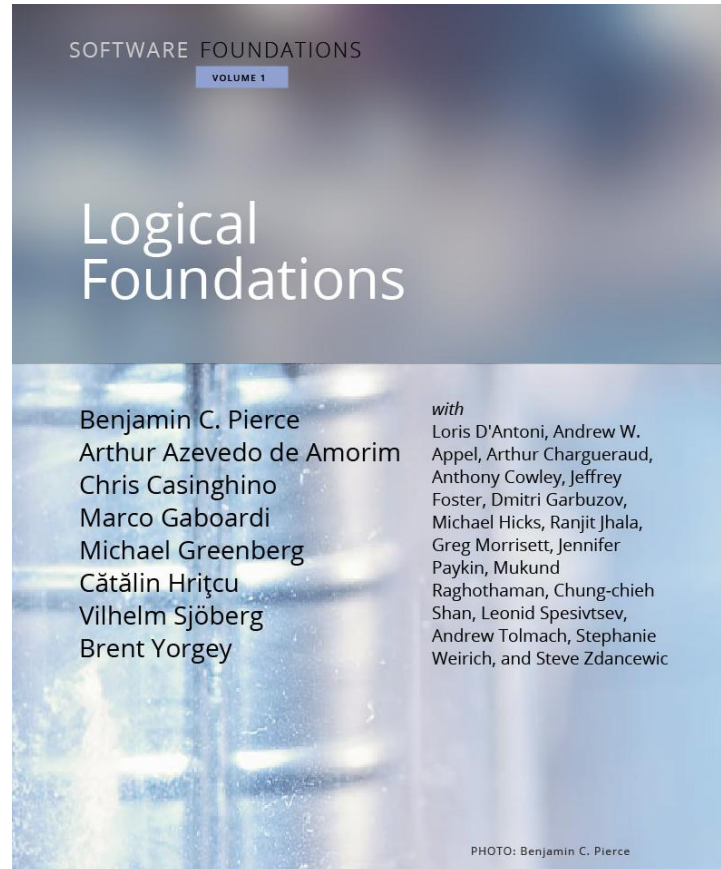
Getting Started with Proofs

- You already know how to write programs: languages, compilers, IDEs, etc.
- How do we write proofs? The same way: with software tools!
- Tool #1: The Rocq proof assistant (<https://rocq-prover.org/>)
- For the first ~5 weeks, we'll learn how to use it to write guaranteed-correct mathematical proofs
- After that, we'll apply those techniques to programs!
- And look at some complicated use cases, esp. *concurrent* programs (see also CS 454)

Interactive Theorem Provers

- In the theorem prover, we can:
 1. Write **definitions**, in a math-like programming language
 2. Write **proofs** about those definitions, using logic “tactics”
 3. See the **proof state** at each point in a proof (what do we know? what do we still need to show?)
 4. Automatically **check** that each step of our proofs is valid

Logical Foundations



- Online textbook, each chapter is a file that can be run in Rocq
- Contents:
 - Introduction to theorem proving
 - Basic logic and functional programming
 - More advanced logic, mostly induction
 - How to describe the behavior of programs

<https://softwarefoundations.cis.upenn.edu/lf-current/index.html>

Getting Started with Rocq

- Distributed via opam, the OCaml Package Manager
- File extension is `.v`
- If it matters, we'll use version 9.0
- Two main IDEs: Visual Studio Code (VSRocq extension) and RocqIDE (available on opam)

Today's Exercise

1. Install opam: [https://ocaml.org/install#linux mac bsd](https://ocaml.org/install#linux_mac_bsd)
If you're using Windows, I strongly recommend using WSL
2. Install Rocq: **opam install rocq-prover**
3. Install VS Rocq (or another IDE if you don't like VS Code)
4. Download and unpack the textbook
(<https://softwarefoundations.cis.upenn.edu/lf-current/index.html>)
5. Run **make** in the textbook's folder to compile the textbook
6. If you finish, run [demo.v](#), then submit a screenshot for Exercise 1/13 on Gradescope (the whole file should be processed). If you haven't finished, submit a description of where you're stuck instead.

If you get stuck, raise your hand, post on Piazza, or come by office hours

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.