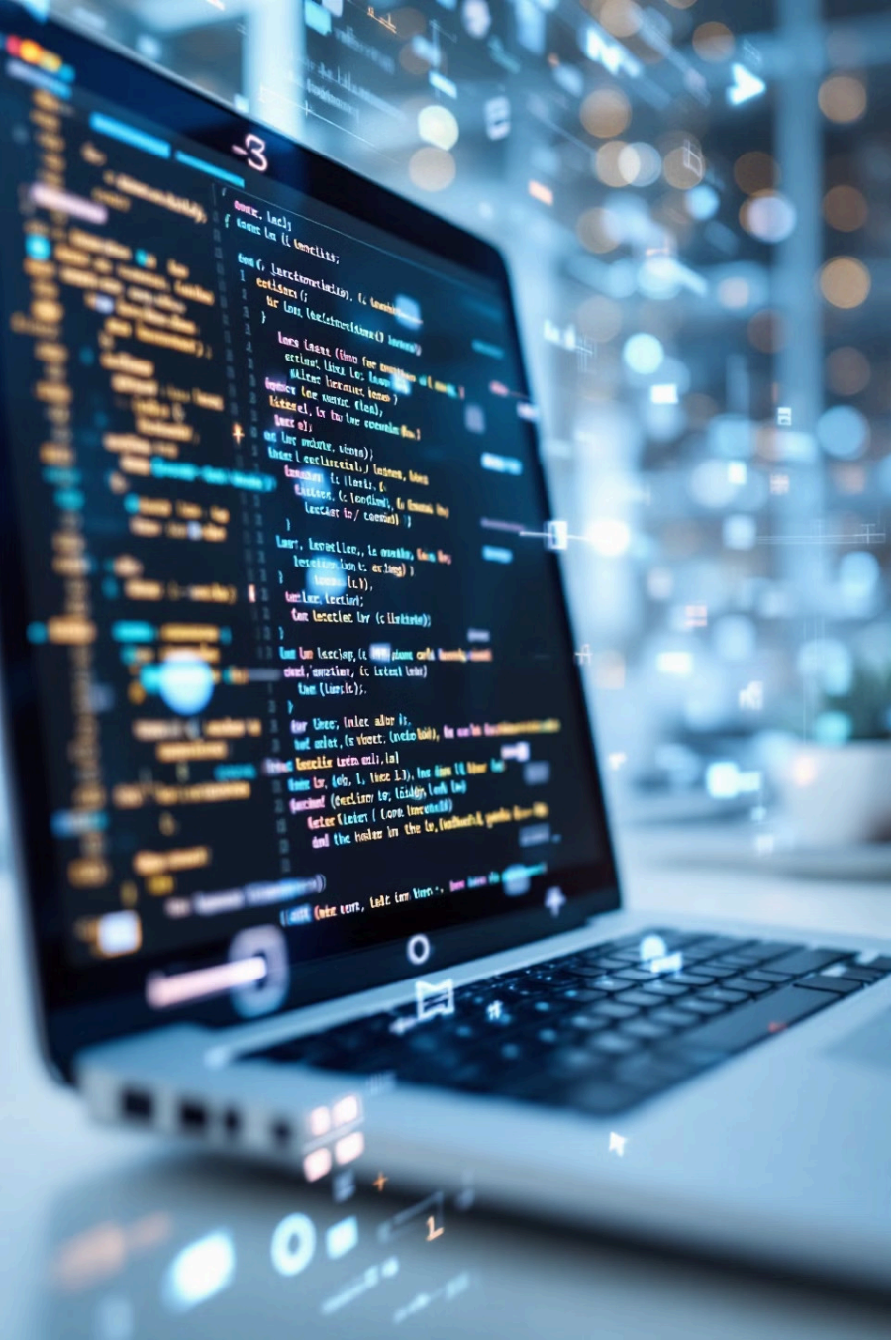


CS292C-1: Computer-Aided Reasoning for Software

Lecture 1: Overview & Motivation

Spring 2025

Instructor: Prof. Yu Feng

by Yu Feng

Who This Course is For

Systems Hackers

Find real bugs in complex systems

Formal Methods Geeks

Prove theorems about program behavior

PL Enthusiasts

Explore solvers and synthesis techniques

About Me

Instructor

Yu Feng

Email

yufeng@cs.ucsb.edu

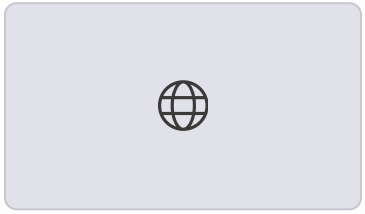
Office Hours

Tue 9-10 AM @ HFH 2157

Research

Programming Languages, Program Verification, Blockchain Security

Why This Course Exists



Software is everywhere

From phones to cars to medical devices

Software breaks all the time

Bugs are inevitable in complex systems

Bugs cost $$$

And sometimes lives

Testing alone isn't enough. We need principled, mathematical techniques.

Famous Software Failures

$370M

Ariane 5

Lost to integer overflow

$4.2B

DeFi Protocols

Drained from Aave bug

Lives

Toyota

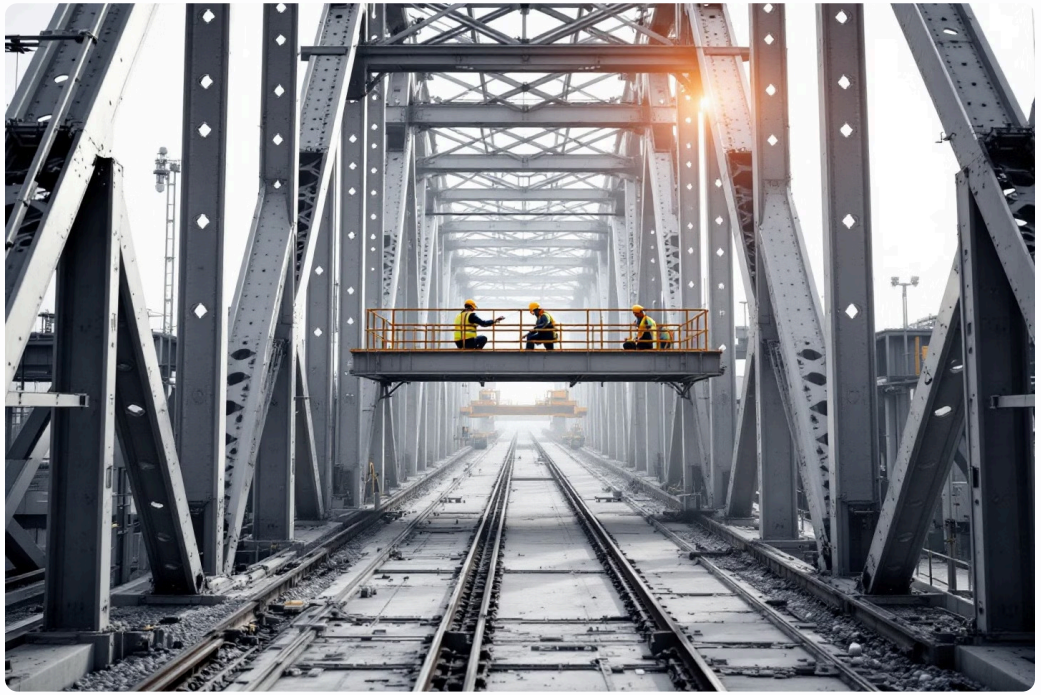
Acceleration bug caused real-life deaths

These weren't bad luck — they were preventable.

How Do We Build Robust Software?

Traditional Engineering

Add steel = make it stronger



Software Engineering

Not so easy...

- Testing ≠ proof
- Code coverage ≠ correctness
- Even defining "correct" is tricky

Key Ideas in This Course

Verification

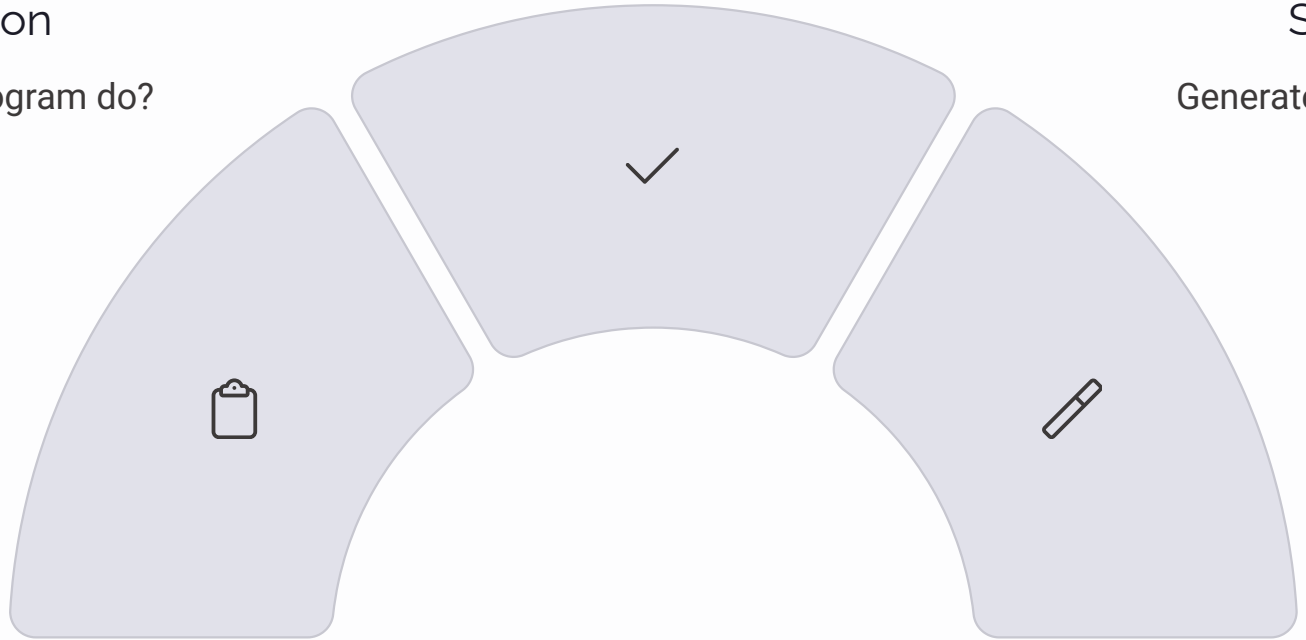
Prove it does what it should

Specification

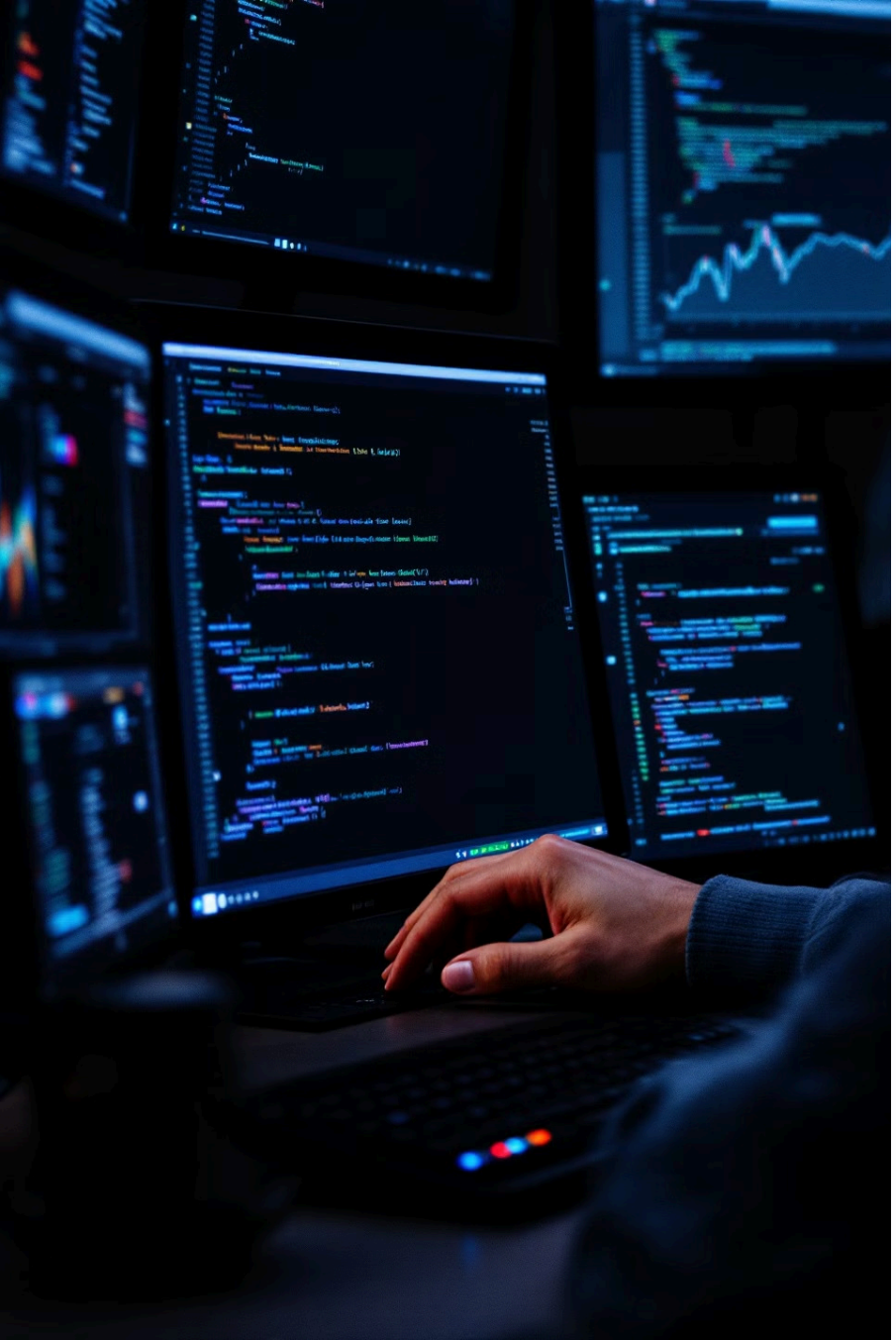
What should the program do?

Synthesis

Generate code from specs



Formal methods let us go from programs → formulas → proofs



Tools You'll Learn About

Hoare Logic

Mathematical reasoning about programs

SMT Solvers

Z3, CVC5 for constraint solving

Symbolic Execution

Analyze code paths systematically

Program Synthesis

Generate code automatically

Some you'll use. Some you'll build.

Course Structure

Topics

- Program semantics
- Hoare logic
- SAT & SMT solving
- Verification condition generation
- Synthesis and solver-aided programming

Work

- 3 Assignments
- 1 Midterm
- Final project/presentation

Prerequisites

Discrete Math

Logic, sets, relations, and proofs

Programming Language Concepts

Semantics, type systems, and language design

Compilers (Nice to Have)

Not required, but helpful

You'll pick up everything else as we go.

Grading & Logistics

Grade Distribution

- **Final Project:** 30%
- **Midterm (no make-up):** 40%
- **Programming Projects:** 30%

Final Project Breakdown:

- Well-documented README: 10%
- Complete, executable codebase: 10%
- Lightning talk: 10%

Course Logistics

- **Q&A:** Slack

Why This Is Exciting

Deeper Understanding

You'll learn to reason like a theorem prover.

Practical Experience

You'll implement pieces of real tools.

Career Opportunities

You'll be ready for research, security, and formal methods jobs.

Let's build software that *actually works*.