ECS 261: Program Verification

Lecture 0 – WQ2026

Welcome!

- This is a **graduate course** in a topic called **program verification**

  (AKA **formal verification**)

  - if you don't know what that is, more on this soon

- Current cap: 50 – we currently have **31 waitlisted**

  Some will drop – more expected off waitlist; depending on size of waitlist, I can request to increase the cap.

Instructor

Your Instructor: Prof. Stanford (Caleb is also OK)

Started at Davis: July 2023

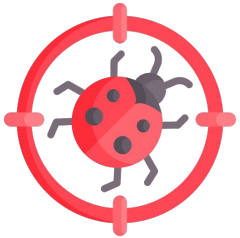
DavisPL Research Group

Plan for today

1. What is this class about?
2. Short discussion/activity
3. FAQ, syllabus, and logistics
4. Demos (time permitting)

What is this class about?

What is this class about?



We live in a world of buggy software

Crashes… Crowdstrike (2024)

CYBERSECURITY

$1.94B in Expected Healthcare Losses Due to CrowdStrike Disruption

CrowdStrike released a root cause analysis of the incident that caused a global outage on July 19

Pietje Kobus

Aug. 13, 2024

CROWDSTRIKE

Crashes… Crowdstrike (2024)

CYBERSECURITY

$1.94B in Expected Healthcare Losses Due to CrowdStrike Disruption

8.5 million devices were confirmed affected by the CrowdStrike outage, but Microsoft says that's only a subset.

CROWDSTRIKE

Crashes… Crowdstrike (2024)

Crowdstrike → device driver → operated in kernel mode (full admin access)

- Crowdstrike driver certified by Windows (WHQL) as being compatible!
- However, driver was written to **load definition files** at runtime – “Agile development” “customers should get the latest protection” – see: Zero day attacks
- Invalid definition file (just contained 0s) - accessed an invalid pointer (0x9c)
  - Definitions can execute arbitrary code and no longer certified!
- Driver marked as a “**boot start driver”** – couldn’t be turned off!

Full explanation here: https://www.youtube.com/watch?v=wAzEJxOo1ts

Crashes… (September 2025)

NBC BAY AREA

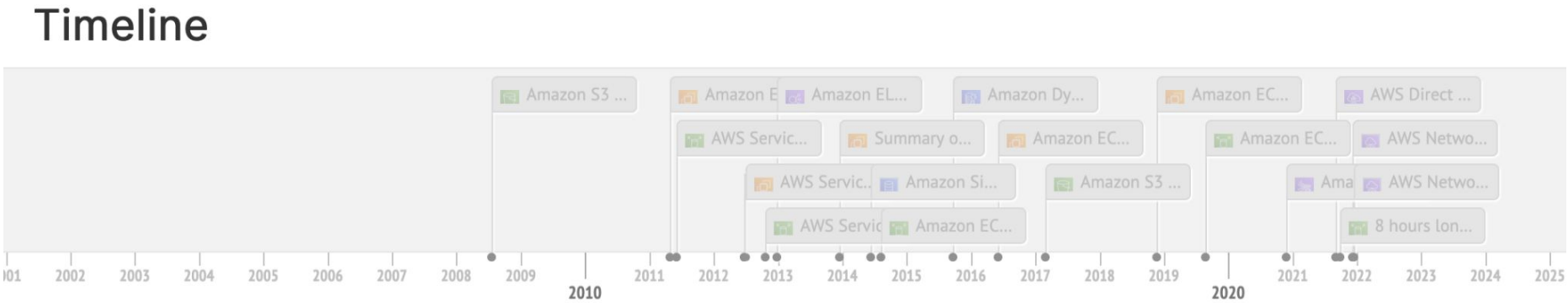
BART

Computer problem halts all BART trains

By Bay City News • Published 2 hours ago • Updated 14 mins ago

NBC Universal, Inc. BART trains were unable to start running early Friday morning after a computer equipment problem shut down the system. Bob Redell, Ginger Conejero Saab and Cinthia Pimentel report.

Outages… AWS (2010 to present)



Source: https://awsmaniac.com/aws-outages/

Outages… AWS (2010 to present)

Timeline

OUTAGES SECURITY & RISK MANAGEMENT

AWS Outage that Broke the Internet Caused by Mistyped Command

Amazon says Tuesday's mayhem resulted from mistake during a routine debugging exercise.

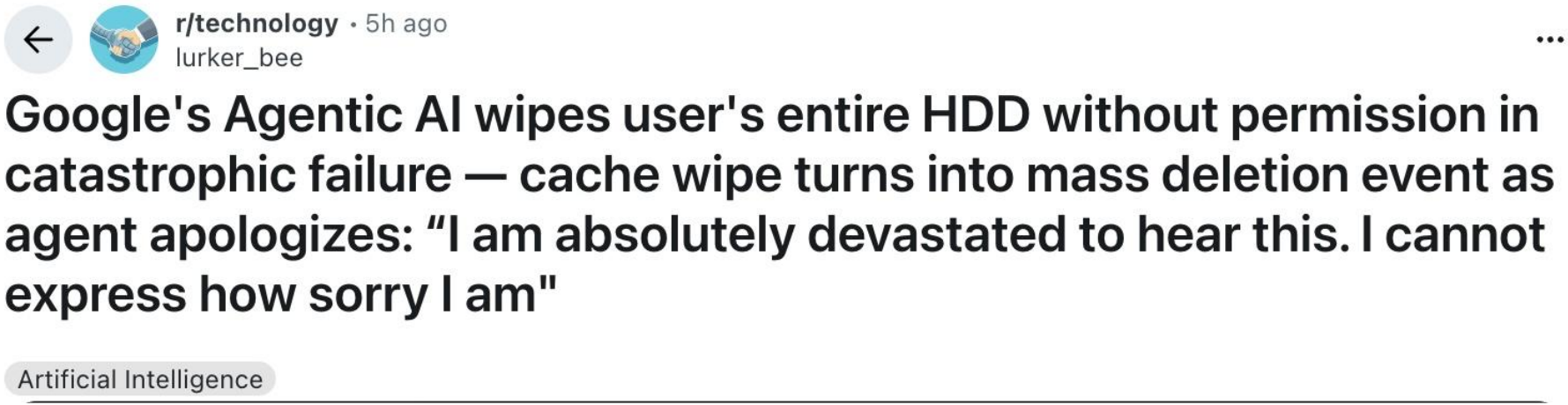
Source: https://awsmaniac.com/aws-outages/

Lost data… MacOS Sonoma

hoakley / March 18, 2024 / Macs, Technology

Serious bug in Sonoma 14.4 will destroy saved versions in iCloud Drive

Lost data… Google AI (December 2025)

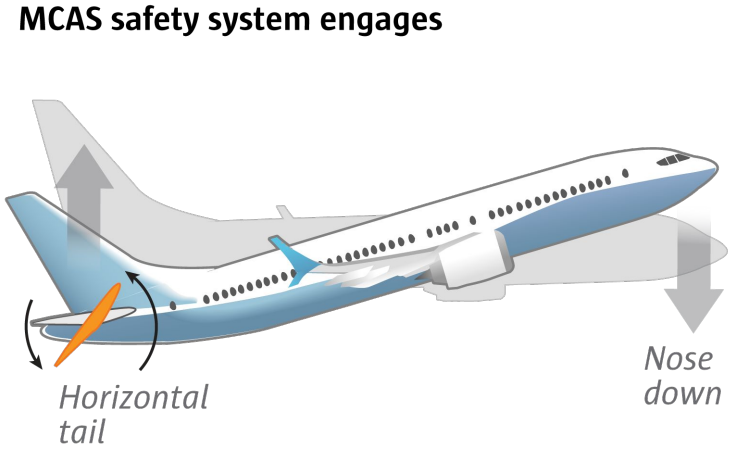


Security bugs… Heartbleed

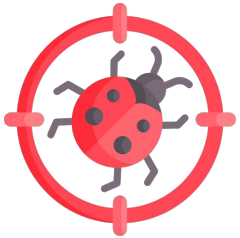
The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to encrypt the traffic, the names and passwords of the users and the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.

And even: Loss of life

Therac-25
(1985-1987, 6 deaths)



Boeing 737 MAX
(2018-2019, 345 deaths)



We live in a world of buggy software

You might have even encountered your own bugs “in the wild”...

File syncing service (Oct 2018)

On Mon, Oct 22, 2018 at 02:33 AM, "Caleb Stanford" <caleb_stanford@alumni.brown.edu> wrote:

•••

> Hello,
>
> I would like to report a bug in which Insync deleted 2 of my files while running solely in the background. I have included information below. After a progressive panic attack trying to find the files, I was finally able to find them in the .insync-trash folder after looking at this page https://help.insynchq.com/resolving-and-reporting-issues/general/retrieving-deleted-or-missing-files.

File syncing service (Oct 2018)

On Mon, Oct 22, 2018 at 02:33 AM, "Caleb Stanford" <caleb_stanford@alumni.brown.edu> wrote:

•••

Hello,

Steps to reproduce:

1. Start with folder A with many files and a folder B somewhere else

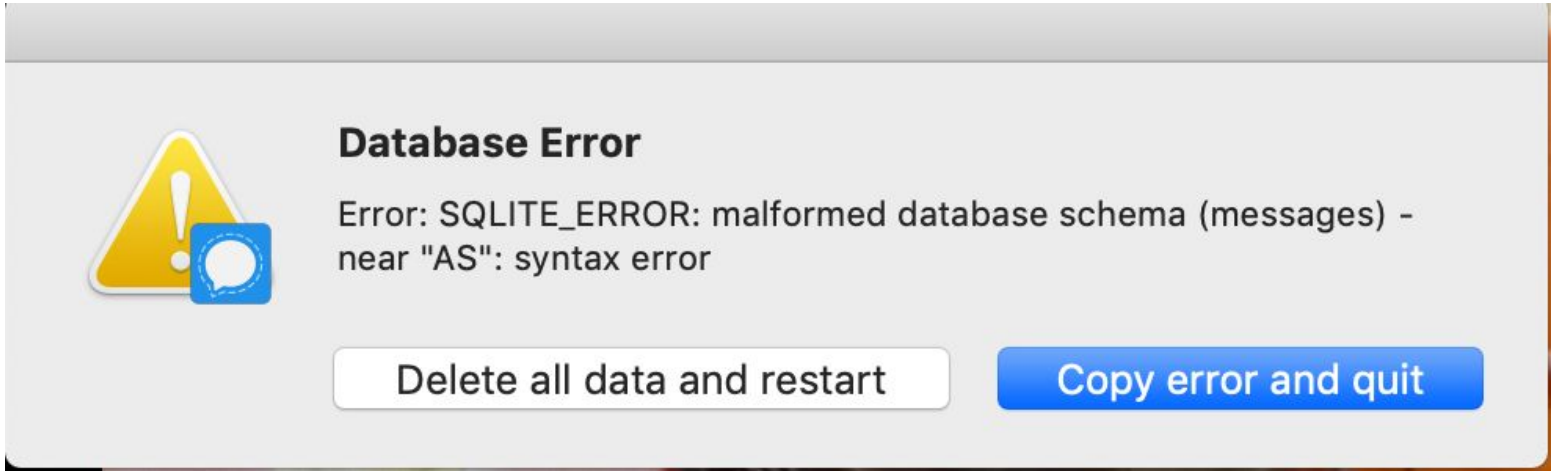
2. Move the files from folder A to folder B by a single cut-and-paste operation

3. While Insync is still processing this move, delete folder A

4. Insync may delete files from folder B, after they are already moved by the system, if the file no longer exists in folder A. The files will be moved to .../.insync-trash/.../B and can be recovered from there.

Social media (May 2021)

Signal messaging app (March 2022)



🙃 (Unknown from January 2019)

AI error (January 2025)

Yes, the **Travelpro Maxlite 5 Rolling Underseat Compact Carry-On** is specifically designed to meet budget airlines' personal item size restrictions. At 15.25" x 13" x 8", it fits within the typical personal item limits of:

- **Frontier Airlines**: 14" x 18" x 8" ✓

Omi

★★★★☆ **Beautiful and compact**

Reviewed in the United States on September 18, 2023

Color: Orchid Pink Purple | **Verified Purchase**

Purchased this for my many trips for work. For me it held enough clothes for a week (i did laundry) and my 2 laptops. It swivele easily thru the airport and for the most part all went well. I normally fly via Frontier and now they have gotten absurd with the drop your carryon into the preformed metal size thingee and this suitcase didn't fit. I told them I had 2 laptops which were unforgiving for squishing in down in size and they instructed me to remove them from the suitcase so I did and it fit comfortab in the size requirement. Then I was instructed that I could not replace them back into the suitcase but needed to "carry them aboard". Stupid... So I walked onto the ramp, opened my suitcase and put them back in. No issue with that. Then I got to my sea and found that with or without the laptops there was NO WAY this suitcase would fit UNDER the seat in front, so I put it top sid So if your thinking it will fit under it will not but it fits easily above. The suitcase is very nice, I will continue to use it for my fligh but now I know not to even try to put it under and to remove my laptops prior to boarding!

17 people found this helpful

Thank you for your feedback. | Report

We live in a world of buggy software

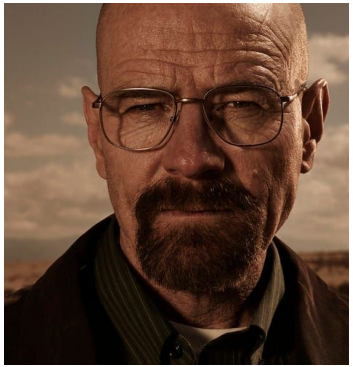
Making the situation worse…

The bug may only show up on **some platforms**

It may require an **esoteric/obscure** input 

Or fail to show up at all.

Heisenbug (n.): a software bug that seems to disappear or alter its behavior when one attempts to study it.



Heisenbugs

Heisenbug (n.): a software bug that seems to disappear or alter its behavior when one attempts to study it.

▲ Ubuntu Bug 255161: Openoffice can’t print on Tuesdays (launchpad.net)

244 points by franze on Aug 13, 2014 | hide | past | favorite | 37 comments

Infinite loop heisenbug: it exits if I add a printout

Asked 9 years, 1 month ago Modified 3 years, 1 month ago Viewed 2k times

Game client (December 2025)

Here's a simple explanation of what's happening: **the client is placing you into two games at once. You join the first game, and then moments later you're placed into a second game.** You play the second game normally, but when the five-minute timer runs out in the first game, that opponent is prompted to make you resign. Once they click it, you're pulled out of the second game and taken to the resignation screen for the first game.

If you start another game, you end up in a third game. At that point, you're now in both the second and third games, and the same cycle repeats.

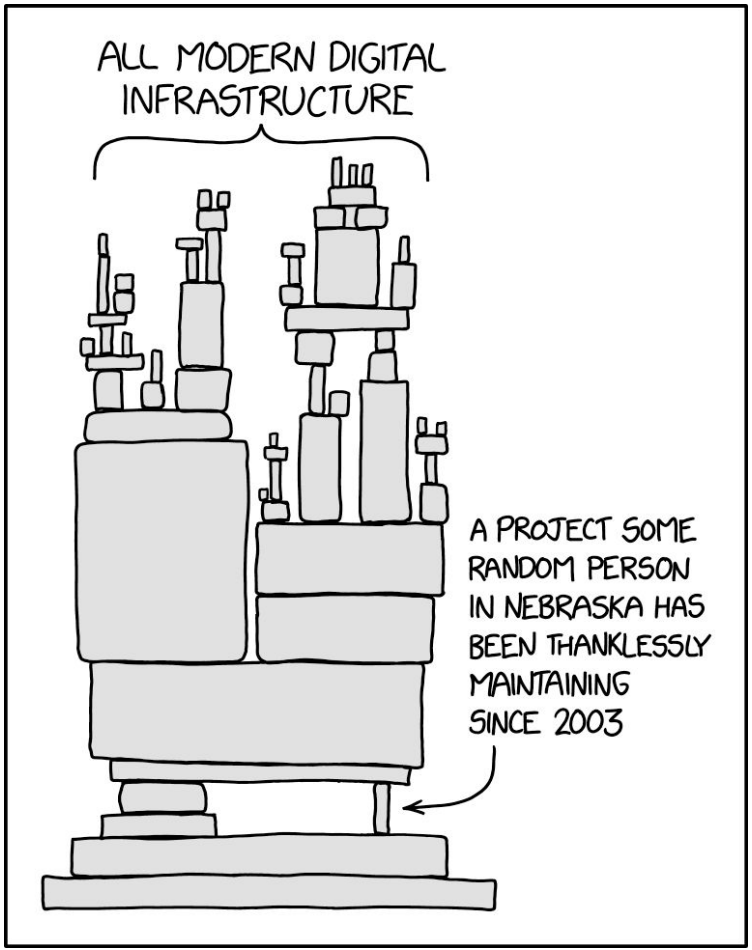
^^(Failure would be studied in a distributed systems class)

Why is software like this?

Why is software like this?

Are we all just terrible programmers?

Why is software like this?



Credit: xkcd

Is there a better way?

Is there a better way?

1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument

Is there a better way?

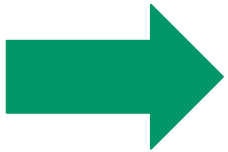
1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument

Z3

Dafny

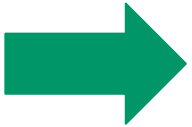
Is there a better way?

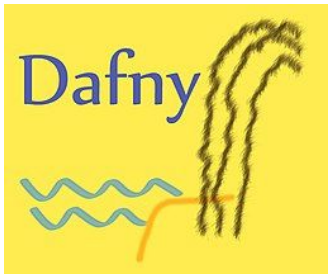
What this class is about



1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument

Real-world tools used in industry





Discussion question

1. Describe what the program **does**
2. Come up with a definition for what this program **should do**
3. Share your definition with your neighbors

```
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Then fill out this poll:

https://forms.gle/2dbFX8P4YBFU9ENA6

Sharing your answers

1. Describe what the program **does**
2. Come up with a definition for what this program **should do**
3. Share your definition with your neighbors

```
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Then fill out this poll:

https://forms.gle/2dbFX8P4YBFU9ENA6

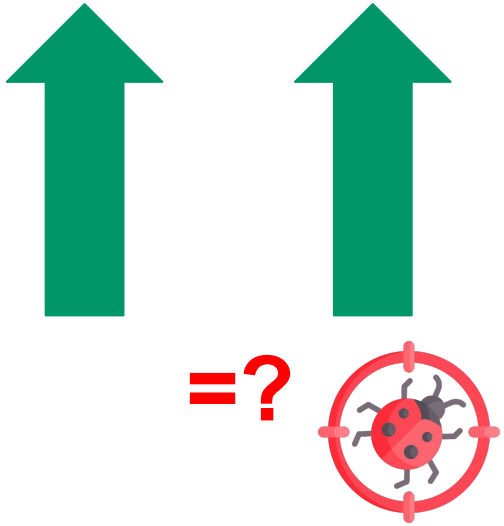
The underlying question: **What is a bug?**

The underlying question: **What is a bug?**

Answer: It Depends!

(Remember that first step)

1. Write down what the program does (and should do)



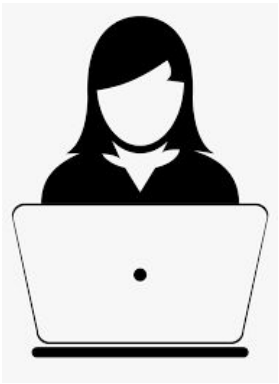
Fundamental question in software development!
Difficult to answer precisely!

Example

Please build me a car

Example

Please build me a car

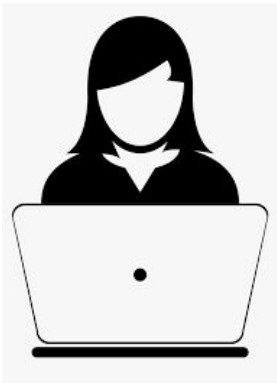


What car? What is the definition you have in mind?

Example

Please build me a car

It should have four wheels and you can drive it places



What car? What is the definition you have in mind?

Example

Please build me a car

It should have four wheels and you can drive it places

Ok, here you go

Example

Please build me a car

… it should also have a roof and seats inside

Ok, here you go

Example

Please build me a car

… it should also have a roof and seats inside

Ok, here you go

Example

Please build me a car

… it should also have a roof and seats inside

… etc.

Ok, here you go

Example 2

Please write a program to check if a number is even or odd

```python
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Ok, here you go

Example 2

Please write a program to check if a number is even or odd

```python
def is_even(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    elif x == 2:
        return True
    elif x == 3:
        return False
    elif x == 4:
        return True
    else:
        return False
```

Ok, here you go

Is this correct?

(Why not?)

Example 2

Please write a program to <do some task>

```
def is_even(x):
    if x == 0:
        retur
    elif x ==
        retur
    elif x ==
        retur
```

Ok, here you go

Is this correct?

(Why not?)

I apologize for the mistake in my previous response. You are correct that the output I provided was not correct. Thank you for bringing it to my attention.

Here is the corrected code to standardize the data:

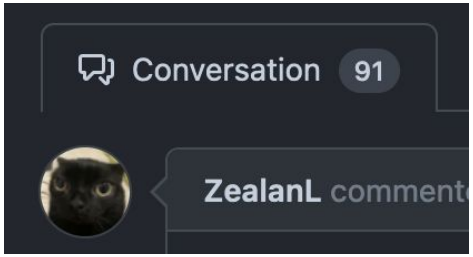
python Copy code

Example 3

Please write a program to play chess

Ok, here you go

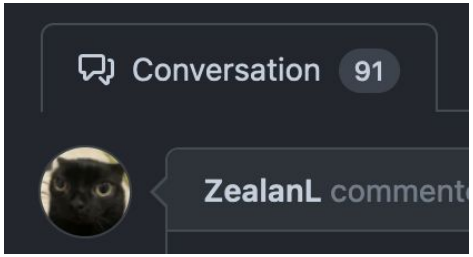
Example 3



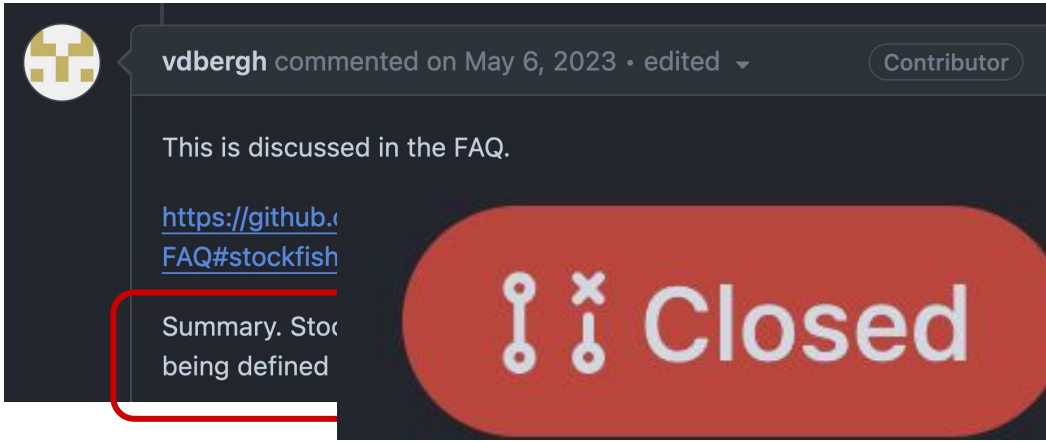
Your program can be used by a bad actor to access and modify arbitrary user memory?

It's a feature, not a bug

Example 3

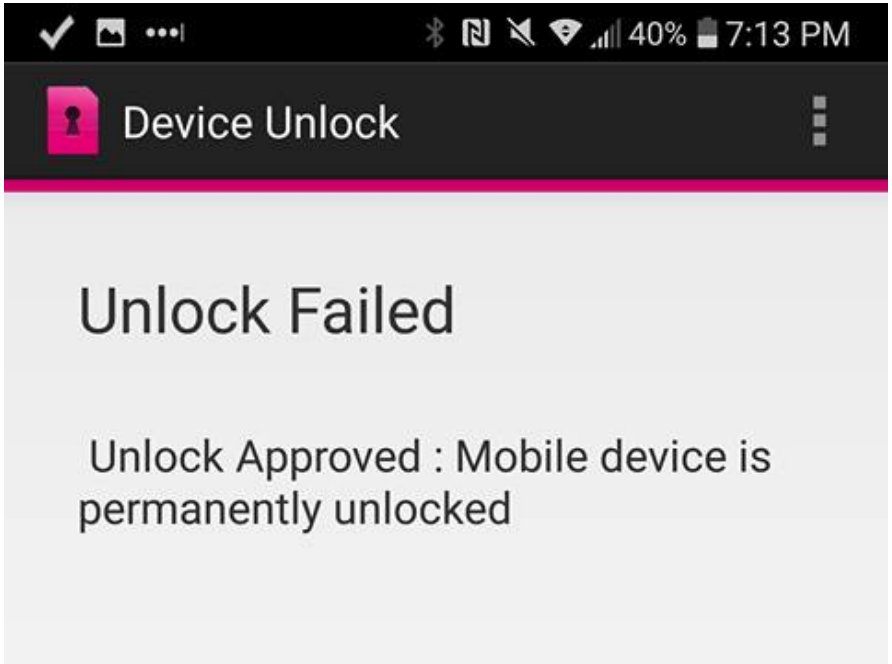


Your program can be used by a bad actor to access and modify arbitrary user memory?



It’s a feature, not a bug

Example 4: is this a bug?



The problem

We need a

clear and unambiguous

way to determine if programs are correct.

The problem

We need a

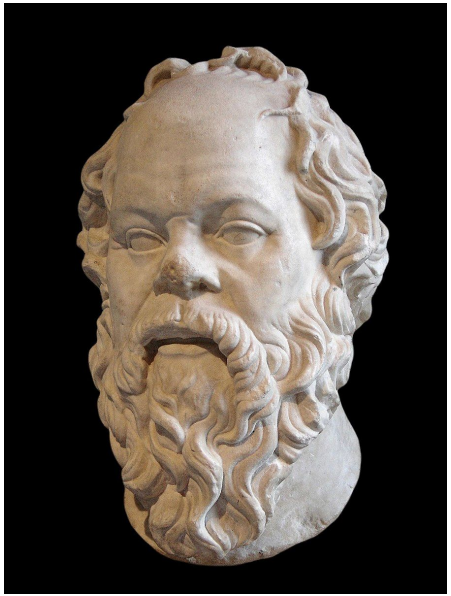
clear and unambiguous

Everyone should agree!

way to determine if programs are correct.

That is:
What the software is;
What it is supposed to do; and
Why it works (or why it doesn't)

Clear and unambiguous?



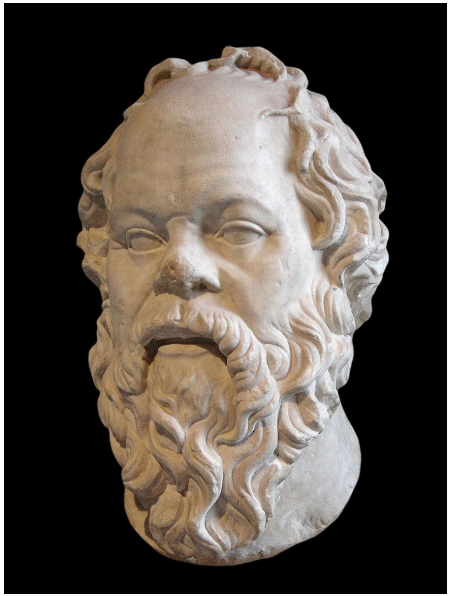
All men are mortal.

Socrates is a man.

Therefore, Socrates is mortal.[2]

Logical Syllogism

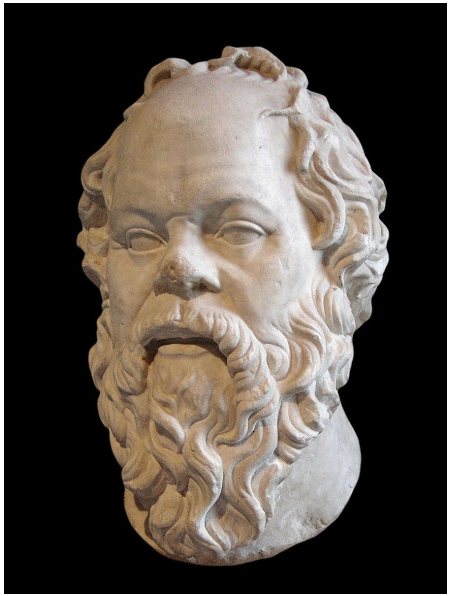
Clear and unambiguous?



A *proof* is a rigorous mathematical argument that demonstrates why a given answer is correct

even to the most serious skeptic.

Clear and unambiguous?



***54·43.** $\vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

[*51·231] $\equiv . \iota`x \cap \iota`y = \Lambda .$

[*13·12] $\equiv . \alpha \cap \beta = \Lambda$ (1)

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda$ (2)

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash .$ Prop

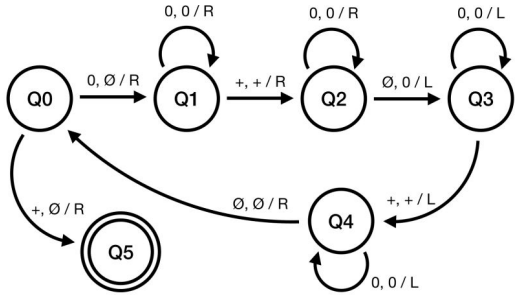
From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

(Don’t worry, I won’t ask you to write this)

Everything is Logic

Proofs can be applied to programs!

Programs are mathematical objects



[0, 1, 2, 3, …]

We can test if the program is correct by (again):

1. Writing down what the program does (and should do)
2. Coming up with a rigorous mathematical argument
3. Using automatic tools check (verify) that argument

⬆️ Program verification in a nutshell

Tools used in industry

Testing tools

(Many others)

Tools used in industry

Automated theorem provers

Microsoft Research

Z3

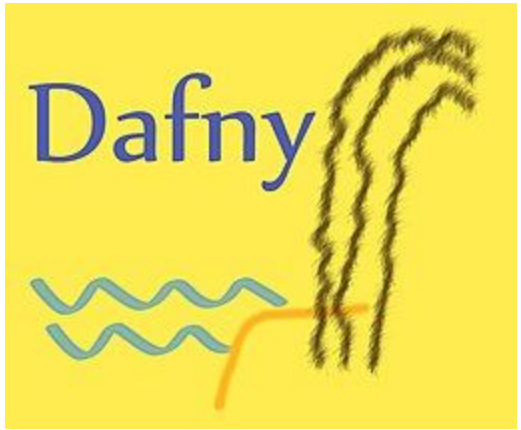
Z3 is a theorem prover from Microsoft Research. It is licensed under the MIT license.

aws Zelkova

"The total number of invocations of Zelkova ranges from a few million to tens of millions in a single day"

Tools used in industry

Program verifiers



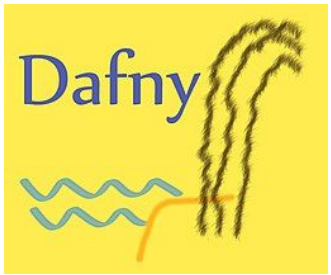
Idris

Agda

(Focus of this class)

Recap: what this class is about

1. Write down what the program does (and should do)
2. Come up with a rigorous mathematical argument
3. Use automatic tools check (verify) that argument



Plan for today

1. What is this class about?
2. Short discussion/activity
3. FAQ, syllabus, and logistics
4. Demos (time permitting)

FAQ

Q: Are there any prerequisites?

A: No formal prerequisites

- A course on writing mathematical proofs at the undergraduate level (e.g. ECS 20/120) is assumed
- I will assume a basic programming background
  - (e.g., ability to write FizzBuzz, etc.)
- A class on mathematical logic (e.g. Phil 112) is helpful, but not required

Q: Will I be required to write mathematical proofs?

A: Yes, somewhat!

- Writing mathematical proofs yourself in important part of program verification!
- In general, the focus of this class is a **hands-on** approach: the tools we will cover can help you check your work

Q: Can I take this class as an undergraduate?

A: Yes

This course assumes no undergraduate background in formal verification tools, so is appropriate for graduate students or **advanced** undergraduates.

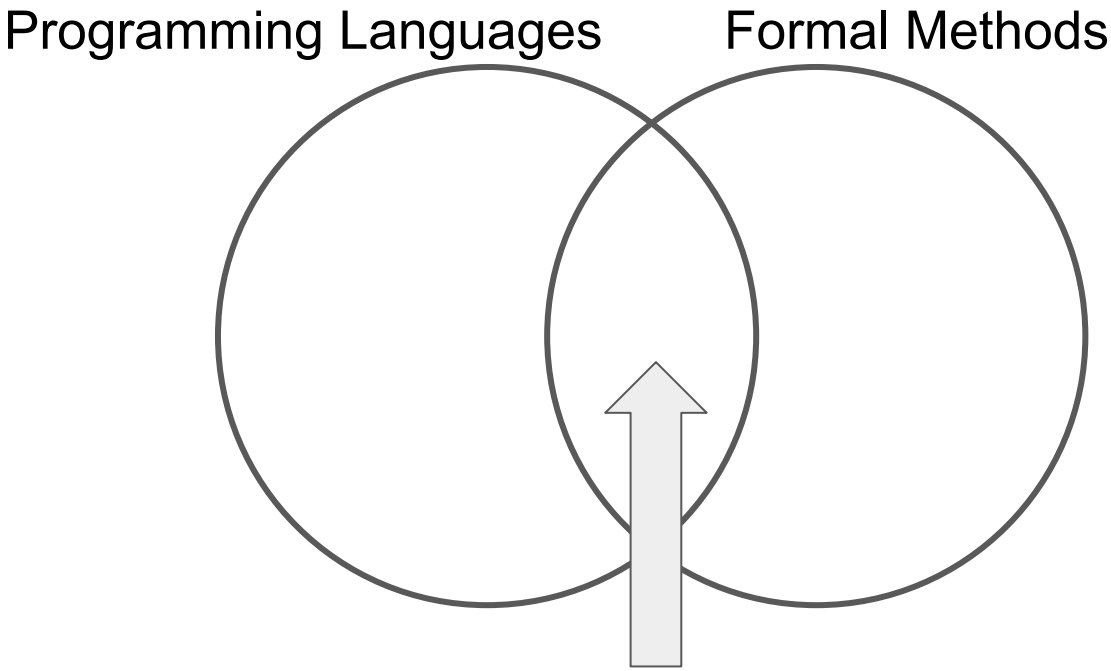
Caveat: you will need to wait 10 days for the drop deadline and make a PTA request; I'll try to get you in, but graduate students may be prioritized for registration, depending on what the department decides.

Q: Can I take this class if I've already taken ECS 189C (special topics / Software Correctness)?

A: Yes (not concurrently)

- As this course assumes no background in formal verification, there will be a **substantial** overlap with 189C
- The final project will be new
  - I will encourage you to focus and go "above and beyond" on the project!

Q: What area of computer science is ECS 261?



Related areas:
theoretical computer science, SE, security, …

Q: Does this course count as a graduation requirement?

A: **Yes,** ECS 261 counts towards your graduation if you are using the **new graduation requirements** approved last year

- **Software bucket** – 3 buckets, 4 units in each bucket

Software	ECS 231	Large-Scale Scientific Computation	4
	ECS 235A	Computer and Information Security	4
	ECS 245	Analysis of Software Artifacts	4
	ECS 260	Software Engineering	4
	ECS 261	Program Verification	4

Q: Is this course right for me?

Short answer: **Probably!**

Long answer: **Especially if:**

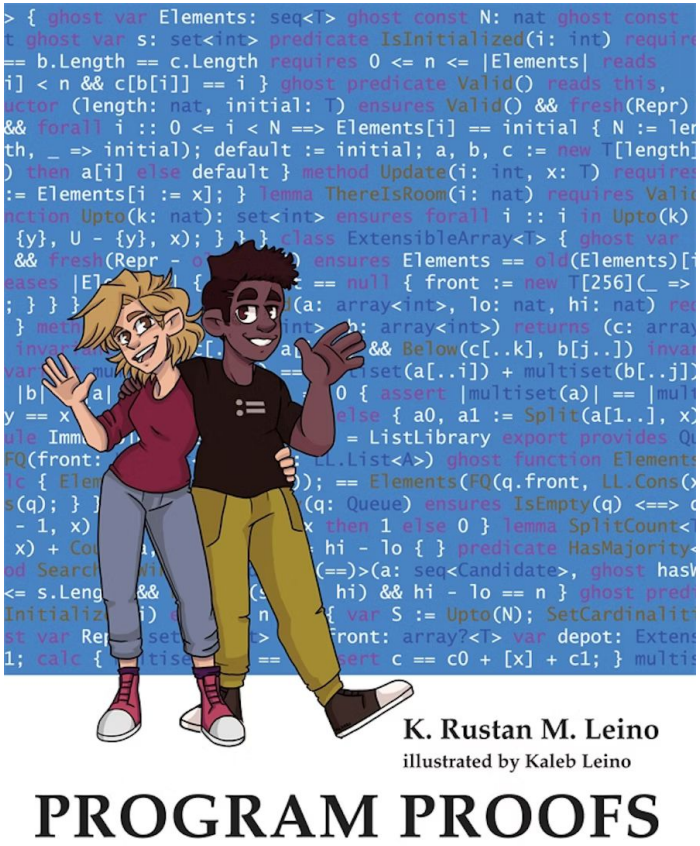
- You want to know the fundamental principles of software verification and learn about tools that are used in industry
- You are interested in thinking mathematical or logically about software and what it does

Syllabus & logistics

Textbook **(optional)**

Program Proofs by Rustan Leino

[MIT Press]



Waitlist

During the first/second weeks: Please do attend the class

- Some people will drop
- I’m not allowed to issue any PTAs (department policy)

End of the second week: let me know if you are still on the waitlist

- Deadline to drop: 10th day of the quarter/instruction (drop deadline)
- Depending on interest, I will request to increase the enrollment cap

TL;DR: Please attend the lectures even if you are on the waitlist.

Graded work

- **Participation (10%)**
- **Homeworks (10%)**
- **Exam (40%)**
- **Final Project (40%)**

Attendance and participation (10%)

Fill out the in-class polls (participation points only)

If you are sick: starting Thursday, you may join the class remotely via Zoom (the quality may not be as good)

If you miss class: Lectures are recorded (best-effort). You can make up the in-class polls at any time

Lectures - live coding

Slides and lecture notes posted before/after the lecture! At:

https://github.com/DavisPL-Teaching/189c

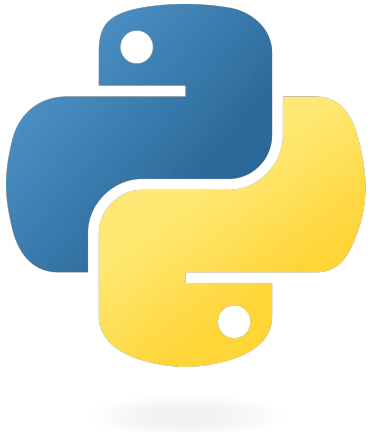
(course links are pinned in Piazza)

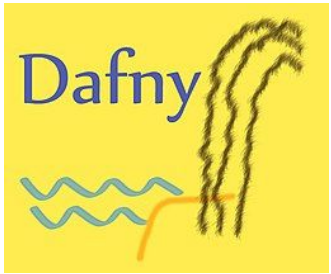
Bring your laptop!

Homeworks (10%)

Roughly bi-weekly problem sets

(About 4 planned)

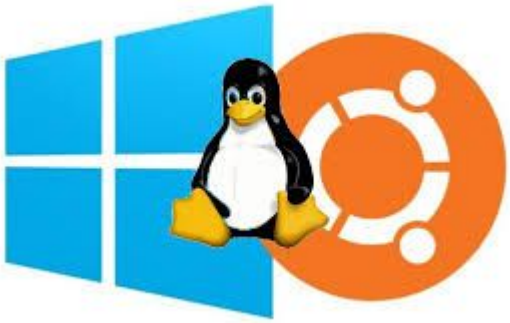




Homework 0: installation help

(submission link posted soon, probably Thu/Fri)

Homework 0: installation help



I recommend using **MacOS or Linux**

If you are on Windows, I recommend

Windows Subsystem for Linux (WSL)

- Well-engineered tool and very useful to know about when doing software development work in the real world!
- https://learn.microsoft.com/en-us/windows/wsl/install

Homework Grading

Most important: please run your code

- Software engineering is about running software!
- Running and testing your code frequently is an important part of developing software in the real world – including ensuring your code works on someone else's machine
- ⚠️ **Code that doesn't run may receive at most half credit. Please double check!**
- ⚠️ **All HW installation issues should be resolved during HW0**

Exam (40%)

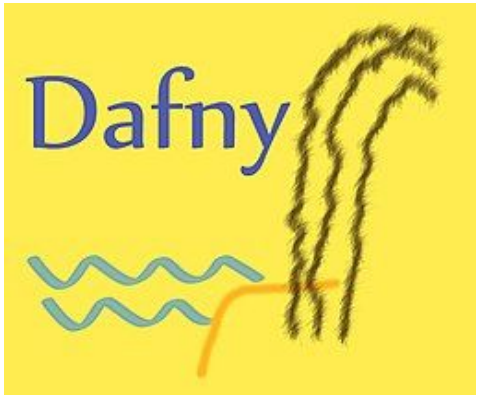
- One final exam
- Please mark your calendar: **Friday, March 20, 6-8pm**
- Closed book, single-sided cheat sheet

Project (40%)

Build a real verified software project in Dafny

Encourage you to apply to your own domain or area of interest!

I will announce more details in class.



Piazza

Please join and monitor Piazza!

ECS 261 on Piazza

- Don’t email me, post to Piazza!
- Make your post public and (if you prefer) anonymous
- Additional platforms: Canvas and Gradescope

AI Policy

AI use is allowed on homework assignments.

Some advice: AI is a tool - use it wisely!

- Use caution that your use of AI is aiding (and not preventing) your own understanding of the material and critical thinking skills.
- (Current generation) AI has some trouble with Dafny and Z3. Don’t fall into the trap of assuming AI is right!
- Exam will be in-class and closed-book
- Advice from Jason Lowe-Power

Collaboration Policy

- Collaboration is allowed and encouraged!
- You can work on the homeworks in groups of 2-3
- Please list your collaborators at the top of your homework
- Everyone should submit their own solution (for the homeworks); group submissions (for the project)

Final project: also in groups, group size TBA! I will try to offer some flexibility

A Rough Schedule (subject to change)

We have 10 weeks, here is a rough plan:

- Week 1: introduction, writing program specifications
- Weeks 2 and 3: logic and logical reasoning tools, Z3
  - Homeworks due
- Weeks 4, 5, 6, 7: deep dive into Hoare logic, Dafny
  - Project proposal: end of week 4 or week 5
- Week 8: advanced topics (if time permits)
- Last 2-3 weeks: final project presentations

Communication

TA: **Lucas Du**

Office hours: TBD (will be posted on Piazza)

Please use Piazza for questions (not email)

Respect and discrimination

Please be nice!

Include everyone in group discussions

Reach out to me if there are any problems

Late policy

Polls and HW0: can be made up

- You don’t need to email me about this, just submit the poll! :)

For HW, I cannot guarantee that late work is graded, but I encourage you to keep working on an assignment if you didn’t finish it by the deadline. Gradescope will allow submissions a few days late, and we may choose to grade these late submissions, at our discretion.

- Why no “hard” policy - this is also how it works in the real world :-)
- **Some homeworks may be long - please start early!**

Other disclaimers

- UC Davis Job Scams Prevention
- Policies against harassment and discrimination
- List of resources for student health and well-being
- SISS for international student issues (and reach out to your graduate advisors)

Questions for me?

Plan for today

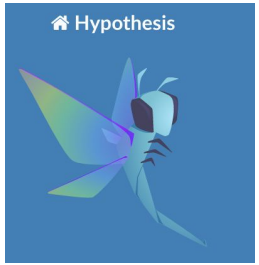
1. What is this class about?
2. Short discussion/activity
3. FAQ, syllabus, and logistics
4. Demos (time permitting)

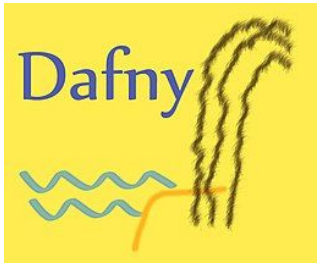
Demos

Lecture 0 Demos

Learning objectives

- Understand the concept of software verification and its importance
- Understand and apply automated verification tools like Z3 for software analysis and logical reasoning tasks.
- Understand and use dedicated program verification tools such as Dafny to develop verified software.
- Understand the formal logic underpinnings of verification tools, and program logics for program reasoning.





✨ Puzzle ✨

I’m thinking of 2 numbers.

The +, *, -, and / of the numbers are (not necessarily in this order):

20, 95, 105, 500

What are the numbers?

✨ Puzzle ✨

(Another one)

The +, *, -, and / of the numbers are (not necessarily in this order):

2, 6, 18, 72

What are the numbers?

✨ Puzzle ✨

(Another one)

The +, *, -, and / of the numbers are (not necessarily in this order):

2, 6, 18, 72

Is this always possible?