COMP3153/9153

Algorithmic Verification

Lecture 1: Course Introduction

Acknowledgement of Country

I would like to acknowledge and pay my respect to the Bedegal people who are the Traditional Custodians of the land on which UNSW is built, and of Elders past and present.

A quick thought experiment

A quick thought experiment

Would you get in a car driven by software written by your fellow students?

Another quick thought experiment

Another quick thought experiment

Can you trust code written by AI?

Who are we?

I am Dr Paul Hunter. My research is on graph theory, algorithms, and formal verification.

Professor Ron van der Meyden will be taking lectures in the second half of term (and in Week 3). He works on formal methods, with applications in distributed computing and computer security; and he developed the MCK model checking tool.

Ronald Chiang, Dao Le, and Ye Li will be taking tutorials.

Dr Liam O'Connor, Dr Rob van Glabbeek, and A/Prof. Peter Höfner are the former lecturers for this course.

Contacting Us

http://www.cse.unsw.edu.au/~cs3153

Forum

There is a discourse forum available on the website. Questions about course content should typically be made there. You can ask us private questions to avoid spoiling solutions to other students.

Administrative questions should be sent to paul.hunter@unsw.edu.au.

Hardware Bugs: 1994 FDIV Bug

$$\frac{4195835}{3145727} =$$

Hardware Bugs: 1994 FDIV Bug

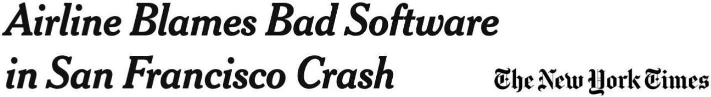
$$\frac{4195835}{3145727} = 1.33370$$

Missing entries in a hardware lookup table lead to 3-5 million defective floating point units.

Consequences:

- Intel image badly damaged
- $450 million to replace FPUs.

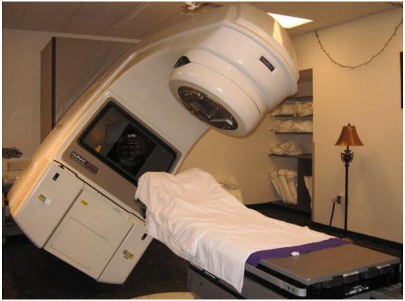
Software Bugs: Asiana 777 Crash in 2014



Airline Blames Bad Software in San Francisco Crash

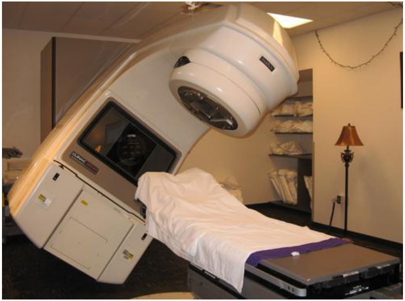
The New York Times

Software Bugs: Therac-25 (1980s)



- Radiation therapy machine.
- Two operation modes: high and low energy.
- Only supposed to use high energy mode with a shield.

Software Bugs: Therac-25 (1980s)



- Radiation therapy machine.
- Two operation modes: high and low energy.
- Only supposed to use high energy mode with a shield.
- Bug caused high energy mode to be used without shield.
- At least five patients died and many more exposed to high levels of radiation.

Software Bugs: Toyota Prius (2005)

- Sudden stalling at highway speeds.
- Bug triggered ”fail-safe” mode (heh).

Software Bugs: Toyota Prius (2005)

- Sudden stalling at highway speeds.
- Bug triggered ”fail-safe” mode (heh).

Consequences:

- 75000 cars recalled.
- Cost unknown. . . but high.

Software Bugs: Ariane 5, Flight 501 (1996)

- Reuse of software from Ariane 4
- Overflow converting from 64 bit to 16 bit unsigned integers.

Software Bugs: Ariane 5, Flight 501 (1996)

- Reuse of software from Ariane 4
- Overflow converting from 64 bit to 16 bit unsigned integers.

Consequences:

- Rocket exploded after 37 seconds.
- US$370 million cost

Northeast Blackout (2003)

- Alarm went unnoticed.
- Bug in alarm system, probably due to a race condition.

Northeast Blackout (2003)

- Alarm went unnoticed.
- Bug in alarm system, probably due to a race condition.

Consequences:

- Total power failure for 7 hours, some areas up to 2 days.
- 55 million people affected
- More than US$6 billion cost

Tesla Recall (Feb 2022)

- Self-driving software would roll through stop signs.
- "Feature" enabled in certain circumstances (30 mph zone, no cars or pedestrians detected)
- Cars will drive through stop signs at up to 6 mph

Tesla Recall (Feb 2022)

- Self-driving software would roll through stop signs.
- “Feature” enabled in certain circumstances (30 mph zone, no cars or pedestrians detected)
- Cars will drive through stop signs at up to 6 mph

Consequences:

- 54,000 vehicles recalled
- Cost: Have you bought a car recently?

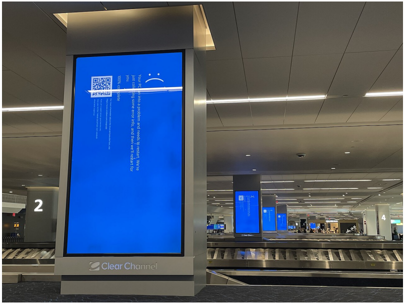
Ethereum bug

What is wrong with this code:

Example

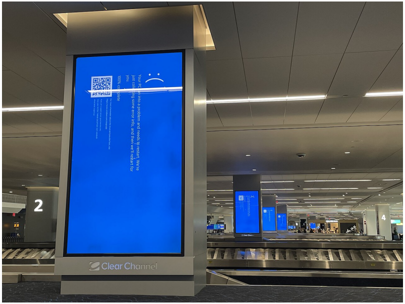
```
transfer(account to, account from, uint amount){
  require (balances[from] > amount);
  balancesFrom := balances[from] - amount;
  balancesTo := balances[to] + amount;
  balances[from] := balancesFrom;
  balances[to] := balancesTo;
}
```

CrowdStrike (July 2024)



- Faulty update to security software
- BSOD and unable to properly restart
- Error was an array index out-of-bounds reference

CrowdStrike (July 2024)



- Faulty update to security software
- BSOD and unable to properly restart
- Error was an array index out-of-bounds reference

Consequences:

- Estimated 8.5 million systems crashed worldwide
- Fix released within hours
- More than US$10 billion cost

Trade-offs in Software Development

Our software should be **correct, safe and secure.**

Our software should be **developed cheaply and quickly.**

Producing safe, secure and correct code

Recently a lot more effort directed towards safer code:

- Rust, Typescript
- Introduction of functional programming principles to imperative languages
- Requirements engineering

Producing safe, secure and correct code

Recently a lot more effort directed towards safer code:

- Rust, Typescript
- Introduction of functional programming principles to imperative languages
- Requirements engineering

What if one bug is too many?

Producing safe, secure and correct code

Recently a lot more effort directed towards safer code:

- Rust, Typescript
- Introduction of functional programming principles to imperative languages
- Requirements engineering

What if one bug is too many?

Can we produce error-free code?

Verification

Ensuring that software or hardware satisfies requirements.

Verification

Ensuring that software or hardware satisfies requirements.

Requirements are:

- That it does what it's supposed to (morally, liveness)

Verification

Ensuring that software or hardware satisfies requirements.

Requirements are:

- That it does what it's supposed to (morally, liveness)
- That it doesn't do what it's not supposed to (morally, safety)

Verification

Ensuring that software or hardware satisfies requirements.

Requirements are:

- That it does what it's supposed to (morally, liveness)
- That it doesn't do what it's not supposed to (morally, safety)

We'll get to more precise definitions later.

Verification

Ensuring that software or hardware satisfies requirements.

Requirements are:

- That it does what it's supposed to (morally, liveness)
- That it doesn't do what it's not supposed to (morally, safety)

We'll get to more precise definitions later.

Talk by Moshe Vardi (70+ year history of Program Verification): `https://www.youtube.com/watch?v=Udajbv263TE`

Does a program satisfy requirements?

We could try testing, but it's not exhaustive.

Does a program satisfy requirements?

We could try testing, but it's not exhaustive.

> *Program testing can be used to show the presence of bugs, but never to show their absence!*

Edsger W. Dijkstra (1970) "Notes On Structured Programming" (EWD249)

Does a program satisfy requirements?

We could try testing, but it’s not exhaustive.

> *Program testing can be used to show the presence of bugs, but never to show their absence!*
>
> Edsger W. Dijkstra (1970) ”Notes On Structured Programming” (EWD249)

We want a rigorous and exhaustive method of verification.

Does a program satisfy requirements?

We could try testing, but it’s not exhaustive.

> *Program testing can be used to show the presence of bugs, but never to show their absence!*
>
> Edsger W. Dijkstra (1970) ”Notes On Structured Programming” (EWD249)

We want a rigorous and exhaustive method of verification.

We also want a method which scales.

Formal Verification

Code

```
...
slowCarDown()

...
```

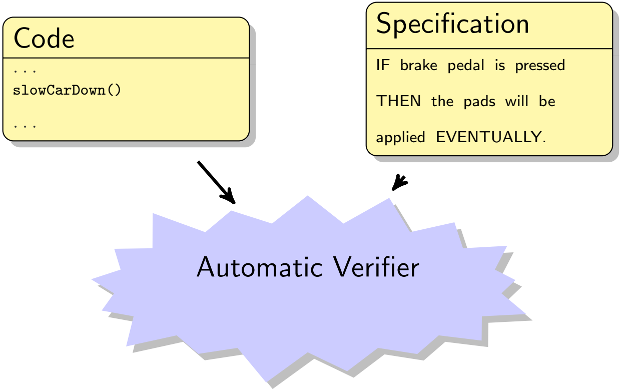
Specification

IF brake pedal is pressed

THEN the pads will be

applied EVENTUALLY.

Formal Verification



Code

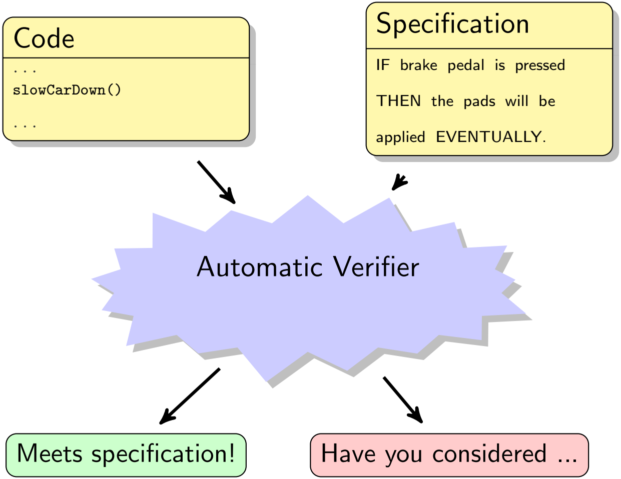
```
...
slowCarDown()

...
```

Specification

IF brake pedal is pressed THEN the pads will be applied EVENTUALLY.

Automatic Verifier

Formal Verification



Code

```
...
slowCarDown()

...
```

Specification

IF brake pedal is pressed

THEN the pads will be

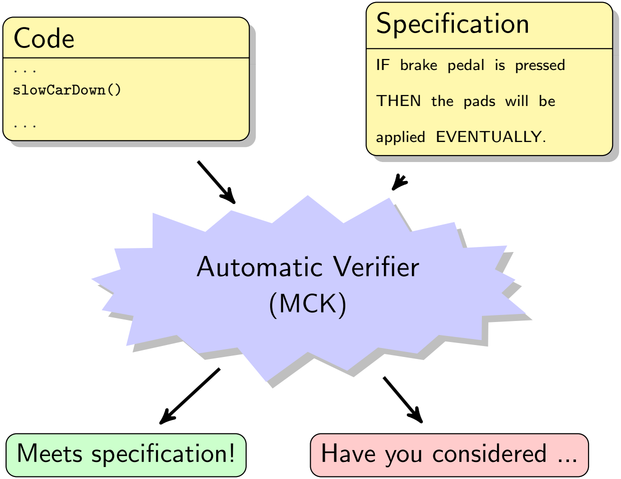
applied EVENTUALLY.

Automatic Verifier

Meets specification!

Have you considered ...

Formal Verification



Code

```
...
slowCarDown()

...
```

Specification

IF brake pedal is pressed

THEN the pads will be

applied EVENTUALLY.

Automatic Verifier
(MCK)

Meets specification!

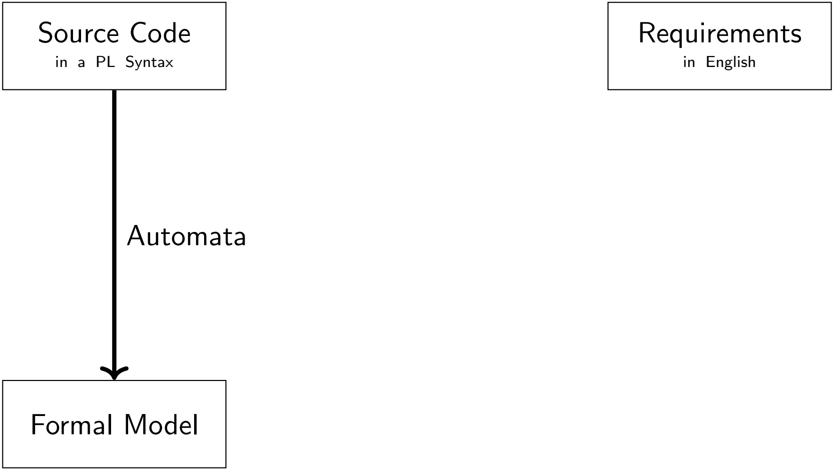
Have you considered ...

Formal Verification (Mathematically)

Source Code
in a PL Syntax

Requirements
in English

Formal Verification (Mathematically)



Source Code

in a PL Syntax

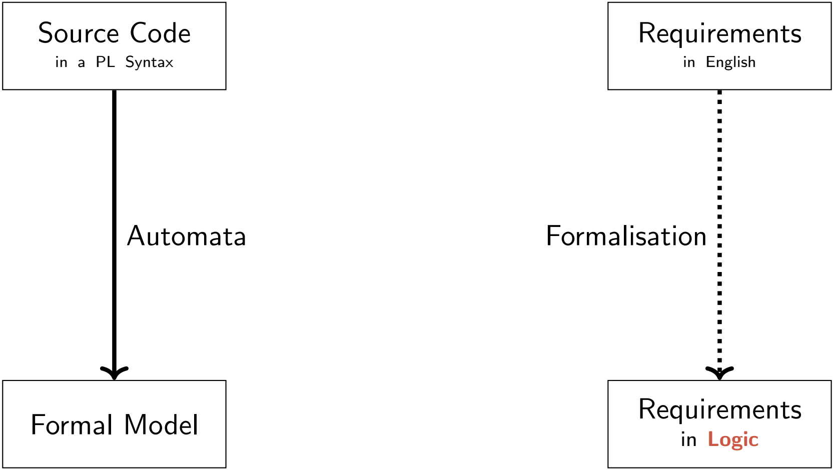
Requirements

in English

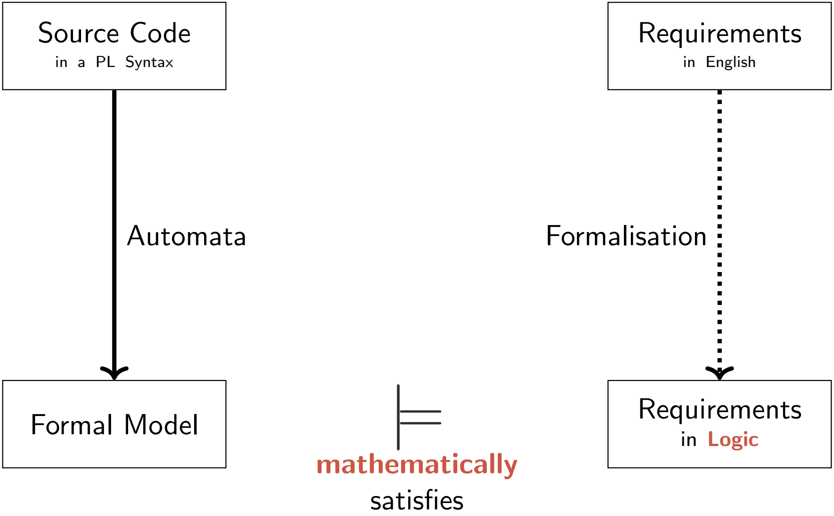
Automata

Formal Model

Formal Verification (Mathematically)



Formal Verification (Mathematically)



Source Code
in a PL Syntax

Automata

Formal Model

Requirements
in English

Formalisation

Requirements
in **Logic**

$\models$

mathematically
satisfies

Learning outcomes

- Develop formal models of software systems, amenable to automatic verification
- Formulate formal requirements for software systems, amenable to automatic verification
- Compare and contrast different algorithms used in automatic verification
- Assess the viability of a number of verification tools for a variety of automatic verification tasks
- Integrate modelling, specification, and verification algorithms to build a formally verified system
- Compare and contrast models, logics, and algorithms used in the verification of timed systems

Course schedule

A (very) tentative course schedule, subject to change:

Week 1	Background, State-based models
Week 2	Automata, Kripke structures, Verification games
Week 3	Tool: MCK
Week 4	Properties, Temporal logics
Week 5	Other logics
Week 6	Flexibility week
Week 7	CTL (Symbolic) Model Checking algorithms
Week 8	LTL Model Checking
Week 9	CEGAR, Bounded Model Checking
Week 10	Epistemic logics, models and verification

What do we expect?

Maths

This course uses a significant amount of *discrete mathematics*. You will need to be reasonably comfortable with *logic*, *set theory* and *induction*. MATH1081 ought to be sufficient for aptitude in these skills, but experience has shown this is not always true.

What do we expect?

Maths

This course uses a significant amount of *discrete mathematics*. You will need to be reasonably comfortable with *logic*, *set theory* and *induction*. MATH1081 ought to be sufficient for aptitude in these skills, but experience has shown this is not always true.

Programming

We expect you to be familiar with imperative programming languages like C. Course assignments may require some programming in modelling languages. Some self-study may be needed for these tools.

Assessment

Assessment in this course consists of:

- weekly formative assessment tasks (presented in the formatif system); and
- a final in-person exam;

with equal weighting between both assessment types.

Formative assessments

- Students select the level of work to be attempted (can be changed)
- Tasks are to be completed to satisfactory level (according to target grade)
- Feedback from teaching staff to achieve task completion
- Final grade determined by portfolio of tasks completed

Formatif system

- 15–30 Tasks in two streams:
  - Stream A: Build a formally verified system
    - Weekly tasks, generally due on Mondays
    - Using tutorials and peer review to refine each component
  - Stream B: Questions addressing abstract concepts
    - Four sets (one per topic)
    - Tailored to target grade
- Submission on Mondays (AOE)
- Feedback/discussion in tutorials
- Must be completed following tutorials

Resources

Lecture Recordings

In previous years, no recordings were made available for this course. I will endeavour make them available this year, however their quality and availability is not guaranteed.

Lectures are intended to be an interactive experience – I will be delivering them in real-time.

The only way to ensure you have the best lecture experience for this course is to attend the lectures!

Resources

Lecture Recordings

In previous years, no recordings were made available for this course. I will endeavour make them available this year, however their quality and availability is not guaranteed.

Lectures are intended to be an interactive experience – I will be delivering them in real-time.

The only way to ensure you have the best lecture experience for this course is to attend the lectures!

Textbooks

This course follows more than one textbook. Each week's slides will include a bibliography. A list of books is given in the course outline, all of the books listed are available from the library.

Tutorial preparation

Given jugs of 3L and 5L, measure out exactly 4L.