Software Verification

Master Programme in Computer Science

Mário Pereira mjp.pereira@fct.unl.pt

Nova School of Science and Technology, Portugal

September 8, 2025

Lecture 1

Before we begin

Meta-language for the lectures: English or Portuguese?

All provided documents are written in English.

Who am I?

- Mário Pereira   `mjp.pereira@fct.unl.pt`
- Office hours: Wednesday at 2pm – office 243, Ed. II
- This is my field of expertise. Do talk to me and put questions!
  - There are many MSc opportunities in this field
  - Some of them with funding

Houston, we have a problem!

Let's watch a video together:

`https://www.youtube.com/watch?v=PK_yguLapgA`

Houston, we have a problem!

Let's watch a video together:

`https://www.youtube.com/watch?v=PK_yguLapgA`

The cause of the disaster? A simple software bug!

Houston, we have a problem!

Let's watch a video together:

`https://www.youtube.com/watch?v=PK_yguLapgA`

The cause of the disaster? A simple software bug!

Too old?

Let's watch a video together:

`https://www.youtube.com/watch?v=PK_yguLapgA`

The cause of the disaster? A simple software bug!

Too old? Try this one:

> *Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken*
>
> Joshua Bloch, Software Engineer (02/06/2006)

Software Verification:

Mathematically Prove that your software is free of bugs!

What about tests?!

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

Edsger W. Dijkstra

How?

The main goal of this course:

- to learn about
- to experiment with
- and master deductive program verification

> *Deductive program verification is the art of turning the correctness of a program into a mathematical statement and then proving it.*
>
> Jean-Christophe Filliâtre

Some major success stories

Flight control software in A380, 2005

safety proof: the absence of execution errors
tool: Astrée, abstract interpretation

proof of functional properties
tool: Caveat, deductive verification

Hyper-V — a native hypervisor, 2008

tools: VCC + automated prover Z3, deductive verification

CompCert — verified C compiler, 2009

tool: Coq, generation of the correct-by-construction code

seL4 — an OS micro-kernel, 2009

tool: Isabelle/HOL, deductive verification

CakeML — verified ML compiler, 2016

tool: HOL4, deductive verification, self-bootstrap

All of these are industrial-scale, formally verified software.

In our case

Learn how to apply deductive verification to small-midsize projects.

1. Reason about functional programs as a mathematical object
2. Attach a logical specification to an imperative program
3. Prove formulae that imply the correctness of the code
4. Verify that heap-allocated data structures are safe

Our proofs: pen-and-paper & using a computer

Functional programming	Imperative programming	Separation Logic
Proofs in Rocq	Proofs in Why3	Proofs in VeriFast

The plan

Here is the plan:

- Midterm: 31st October (Friday)
- Final test: 28th November (Friday)
- Handout 1: 4th October (Saturday)
- Handout 2: 8th November (Saturday)
- Handout 3: 4th December (Wednesday)

Nothing of the above is yet official!!!

Handouts:

- A functional algorithm or data structure verified in Rocq
- An algorithm involving mutable state verified in Why3
- A heap-dependent data structure verified in VeriFast

Evaluation components:

1. *teórico-prática* (TP): one midterm ($T1$) + one final test ($T2$)
2. *prática* (P): three handouts ($HO1-3$, groups of two)
3. final exam (Ex)

Evaluation formula:

$$\begin{aligned} P &= (HO1+HO2+HO3)/3 \\ TP &= (T1+T2)/2 \quad \text{OR} \quad Ex \\ F &= TP, & \text{if } TP < 9.5 \\ F &= 0.7\,TP + 0.3P & \text{if } TP \geq 9.5 \end{aligned}$$

Frequência:

$$\begin{aligned} TP &\geq 9.5 \\ P &\geq 9.5 \end{aligned}$$

> *I hear and I forget. I see and I remember. I do and I understand.*
>
> *Confucius*

Simply want to succeed? Lecture notes and exercise sheets are enough.

Want to excel? Go beyond lectures and lab sessions!

- read the books
- do more exercises on your own
- talk to the teacher
- use the Slack workspace

Let's avoid the passive attitude during lectures:

- I will write a lot on the board. Bring your notebook.
- I will do many live demos. Bring and use your laptop.

That said, I am planning on organizing many extra hands-on sessions.

Functional Programming (1/3)

1. extensional equivalence
2. referential transparency
3. reasoning about functional programs
   - simplification
   - rewriting
   - case analysis

- Software Foundations, Volume 1:
  `https://softwarefoundations.cis.upenn.edu/lf-current/index.html`
  (Chapter 1)