Proofs are Programs

Summer 2023 @ Ruhr Uni Bochum

Lecturers: Cătălin Hrițcu and Clara Schneidewind

TAs: Sebastian Holler and Maxi Wuttke

Max Planck Institute for Security and Privacy (MPI-SP)

This course in a nutshell

1. Logic and proofs
2. Functional programming
3. Program verification

- Using the Coq proof assistant
- Curry-Howard correspondence
  - proofs = programs
  - bridge between logic and computer science

["Le coq mécanisé" picture by Lilia Anisimova]

Logics and proofs

Q: How do we know something is true?

A: We prove it

Q: How do we know that we have a **proof**?

A: We need to know what it means for something to be a proof.
First cut: A proof is a "logical" sequence of arguments,
starting from some initial assumptions

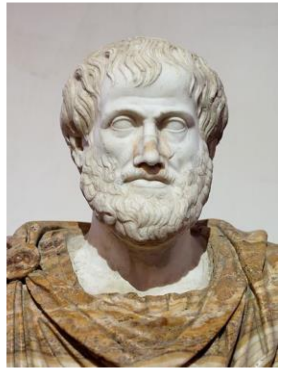
Q: How do we agree on what is a **valid** sequence of arguments?
Can any sequence be a proof? E.g.

All humans are mortal

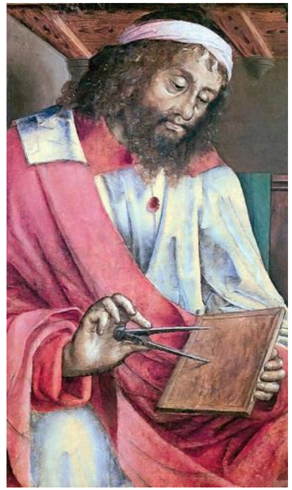
All Greeks are human

Therefore I am a Greek!

A: No, no, no! We need to think harder about **valid** ways of reasoning...



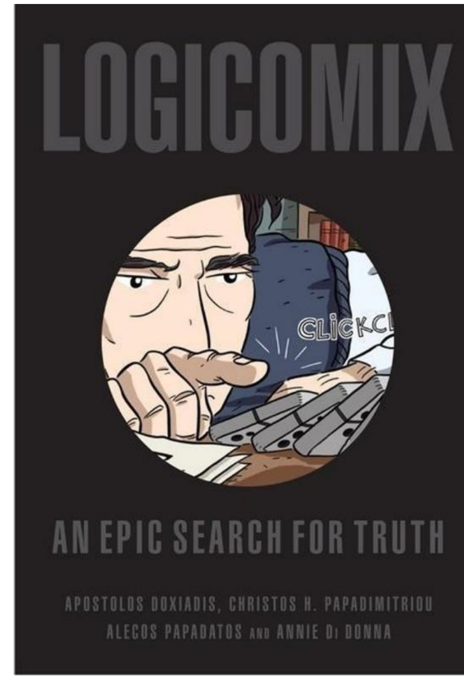
Aristotle
384 – 322 BC



Euclid
~300 BC

The story of logics is quite fascinating

- **David Hilbert** (1862 – 1943)
- **Gottlob Frege** (1848-1925)
  - first-order logic (∀x. P x /\ R x x)
  - derive the laws of arithmetic from first principles (Problem #2)
- **Bertrand Russell** (1872 - 1970)
  - paradox, inconsistency in Frege's logical system, tried to fix it
  - Principia Mathematica
- **Kurt Gödel** (1906 - 1978)
  - incompleteness theorems



- **Gerhard Gentzen** (1909-1945) -- consistency of arithmetic
- **Alonzo Church** (1903 - 1995) -- lambda calculus, simple theory of types
- **Alan Turing** (1912 - 1954) -- undecidability of arithmetic, halting problem
- ...

Logics and proofs

- Foundation of **mathematics** and **computer science**
  - **formal proofs** with respect to **valid inference steps/rules**
- **This course: typed constructive higher-order logic**
  - **higher-order**
    - can quantify not only over individuals (∀x:nat. P x)
    - but also over propositions (∀P:Prop. P),
      predicates (∀Q:nat->Prop. Q x), relations (∀R:nat->nat->Prop. R x y),
      functions (∀f:nat->bool, ∀g:(nat->bool)->bool), ...
  - **constructive**, aka **intuitionistic logic**:
    - a proposition is valid if one can construct a proof
    - philosophically rejects excluded middle (P ∨ ¬P, classical logic)
  - **typed** (dependently typed)

Logics and computer science

- **Logics and CS greatly influenced on each other**, e.g.:
  - **automated theorem provers** (e.g., SAT and SMT solvers)
  - **model checkers** (the course right after this one!)
  - **proof assistants**: Coq, Isabelle, HOL family, F*, ACL2, etc.
    - interactively constructed, machine-checked proofs
    - addictive, gamification of proofs
- This course: **Coq proof assistant**
  - developed at Inria since 1983 (in OCaml)
  - Curry-Howard: proofs = typed purely functional programs

Functional programming λ

- **Try to write computations as pure functions**
  - **without side-effects**, such as mutating the heap
    - sorting a list in place (imperative) vs producing a new list (functional)
  - **Coq is purely functional = zero side-effects**
    - all computations are mathematical functions (in particular terminating)
  - **Functional programming languages** like OCaml, Haskell, ...
    - try to reduce and/or control side-effects
    - make it easy to write pure functions

Functional programming in practice

- **Functional languages have some practical success**
  - **Facebook** (OCaml, Haskell), **Microsoft** (F# and F*), **Twitter** (Scala)
  - **Financial industry**: Jane Street (OCaml), various banks (Haskell, ...)
  - **Blockchains**: Cardano (Haskell, Rust), Tezos (OCaml), ...
- **Not yet fully mainstream, but ...**
  - **Functional programmers earn more** (Stack Overflow survey)
  - **Many ideas already adopted by mainstream languages**: lambdas, generics in Java/C#, Google's Map-Reduce, ...
  - **Makes program verification and informal reasoning easier**

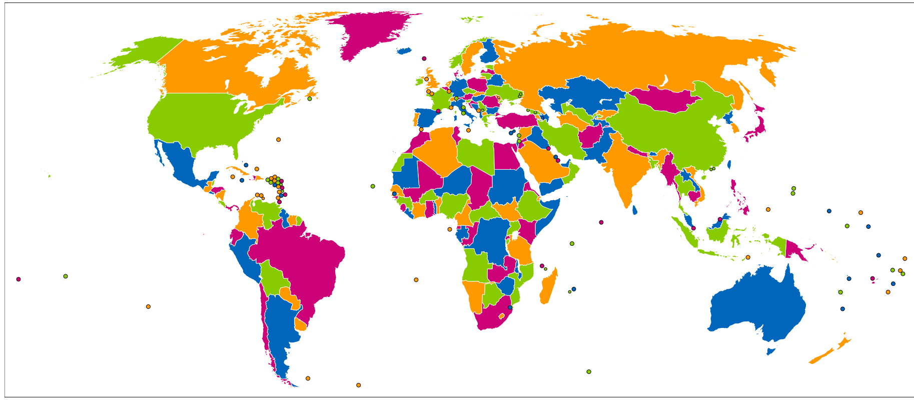
λ

Program verification in proof assistants

Proving mathematically that a **program** satisfies a **specification**

- **verification of smart contracts** (Coq, F*, ...) -- Clara working on this hot topic
- **the CompCert C compiler** (Coq) -- Catalin working on a more secure version
- **the seL4 OS microkernel** (Isabelle/HOL), **the CertiKOS hypervisor** (Coq)
- **EverCrypt cryptographic library** (F*) -- shipping in Firefox and the Linux kernel
- **EverParse3D verified binary parsers** (F*) -- shipping in Windows kernel
  - Catalin working on design of F* proof assistant ("Coq + SMT automation")
- **Libjade high-assurance post-quantum crypto library** (EasyCrypt)
  - Peter Schwabe and Gilles Barthe from MPI-SP

Machine-checked proofs of math theorems



[World map from Wikipedia]

- **The 4-color theorem**
  - proved in 1976, simplified in 1996
  - proofs rely on a computer
  - infeasible for a human to check
  - initially not accepted by all mathematicians
  - mechanized in Coq by Georges Gonthier in 2005
- **Odd order theorem (Feit-Thompson)**
  - Coq proof from 2012 introduced the Mathematical Components library
- **Construction of perfectoid spaces** (Peter Scholze, Lean)
  - Mathlib math library for Lean has over 100,000 theorems, over 1,000,000 lines of code, and an active community including many mathematicians

This course

- **Write purely functional programs in Coq**
  - natural numbers, lists, maps, trees, program syntax
- **Verify these programs by proving theorems about them**
  - case analysis, induction, inversion, proof automation, ...
- **Curry-Howard correspondence**
  - proofs = typed purely functional programs

- **Advice: focus, ask questions, interact**

Logical Foundations

Benjamin C. Pierce
Arthur Azevedo de Amorim
Chris Casinghino
Marco Gaboardi
Michael Greenberg
Cătălin Hrițcu
Vilhelm Sjöberg
Brent Yorgey

with
Loris D'Antoni, Andrew W. Appel, Arthur Charguéraud, Michael Clarkson, Anthony Cowley, Jeffrey Foster, Dmitri Garbuzov, Olek Gierczak, Michael Hicks, Ranjit Jhala, Ori Lahav, Yishuai Li, Greg Morrisett, Jennifer Paykin, Mukund Raghothaman, Chung-chieh Shan, Leonid Spesivtsev, Caleb Stanford, Andrew Tolmach, Philip Wadler, Stephanie Weirich, Li-Yao Xia, and Steve Zdancewic

READ

DOWNLOAD

This is the version used in the Proofs are Programs course given at Ruhr University Bochum in summer 2023 (based on Logical Foundations Version 6.3, 2023-04-05 16:10, Coq 8.16 or later)

Textbook written in Coq; our version at:

https://mpi-sp-pap-2023.github.io/book

Lecture logistics

- 12 lectures: first 6 by Clara, last 6 by Catalin
- Public holidays, so no lecture on 18 May and 8 June
- Vacation 29 May to 2 June, so no lecture, no tutorials
- ~~Starting next week only in presence~~
  - Since some participants asked we decided to still record and stream the following lectures (announcement will follow)

Exercises

- **This is a very hands on course**
  - Coq turns proofs & logic into programming, and it's fun!
- **New exercise sheet will be released after most courses**
  - 1(-2) Coq file(s) with holes you have to fill (basically 1-2 book chapters)
  - so there will be 8-10 exercise sheets in total
- You have to **turn in your solution before next course**
  - up to Wednesday at 23:59
  - Easter exception: this week's exercise sheet is due in 2 weeks
- **We'll be using Moodle** (probably for everything)
- **Exercises count for 40% of final grade** (+5% in bonus points)
  - not required to do the optional exercises; they don't count for grade
- **Exercises are individual, please don't share your solutions in any way!**
- **Exercises highly recommended even if you're not taking this for credit**

Tutorials: Q&A about the exercise sheets

- **TAs:** Sebastian Holler and Maxi Wuttke
- **Monday 12:00-14:00** and **Tuesday 12:00-14:00**
- You can come to any/both and ask questions
- You can also work on your own during tutorials
  - and ask questions as they arise
- If you manage to solve all the exercises and prefer to not come, no problem, you are not forced to come
  - can sometimes still get good advice though
  - can also ask about old assignments (but solutions anyway on Moodle)
- Seems okay to bring your lunch / arrive late / leave early

Exams

- **Midterm exam** (optional)
  - practice for the final exam
  - also written, on paper
  - duration: 60 minutes
  - bonus points: up to 10%
  - date: 23 May, 13:00-14:00

- **Final exam**
  - written, on paper
    - so we will also teach you how to write down proofs informally
  - duration: 120 minutes
  - 60% of the grade
  - date: 3 August
  - re-exam: 19 September

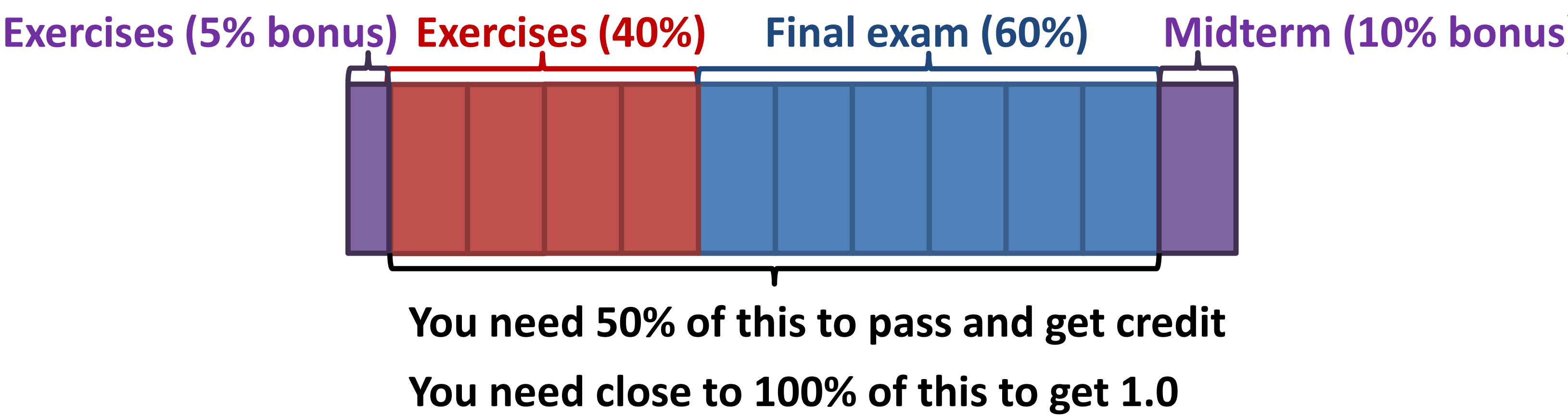
Final grade

CASA PhD students

– attendance requirement is to receive 50% of the exercises score

You can get credit if you study

- Computer Science BSc (soon MSc too)
- IT Security BSc and MSc
- Mathematics MSc



Already considering follow up course (next semester?)

Foundations of Programming Languages, Verification, and Security