CS-428

Interactive theorem proving

Barrière Aurèle, Pit-Claudel Clément

Cursus	Sem.	Type
Computer science minor	E	Opt.
Computer science	MA2, MA4	Opt.
Cybersecurity	MA2, MA4	Opt.
Data Science	MA2, MA4	Opt.
SC master EPFL	MA2, MA4	Opt.

Language of teaching	English
Credits	6
Session	Summer
Semester	Spring
Exam	During the semester
Workload	180h
Weeks	14
Hours	**5 weekly**
Lecture	2 weekly
Exercises	1 weekly
Labs	2 weekly
Number of positions	

Summary

A hands-on introduction to interactive theorem proving, computer-checked mathematics, compiler verification, proofs as programs, dependent types, and proof automation. Come learn how to write computer-checked proofs and certified bug-free code!

Content

- Intro to the Coq proof assistant (logic, higher-order functions, tactics)
- Functional programming (inductive types and fixpoints)
- Structural induction (data structures and verified algorithms)
- Interpreter-based program semantics (intro to compiler verification)
- Inductive relations (predicates, rule induction)
- Automation and tactics I (bottom-up reasoning and logic programming)
- Operational program semantics (small-and big-step semantics)
- Program logics (hoare triples)
- Automation and tactics II (top-down reasoning)
- Type systems (Simply-typed lambda calculus)
- Dependent types and equality proofs
- Automation and tactics III (proofs by reflection)
- Real-world theorem proving (various topics)

Learning Prerequisites

Recommended courses

This course assumes no knowledge of programming language theory. The following courses may be useful, but are not required:

- CS-320 Computer language processing (to introduce the concept of interpreter)
- CS-425 Foundations of software (to introduce type systems and the lambda calculus)
- CS-550 Formal verification (for a different perspective on theorem proving)

Important concepts to start the course

- Functional programming

Learning Outcomes

- Implement purely-functional algorithms in the Gallina language
- Translate informal requirements about software into precise mathematical properties
- Plan and carry out mechanized proofs in Coq (e.g. maths, algorithms, compilers, type systems)
- Automate repetitive proof tasks by crafting simple custom decision procedures

Teaching methods

- Lectures
- Live-coding sessions

Expected student activities

- Lectures
- Programming and verification assignments
- Project (proposal, check-in, presentation, report)

Assessment methods

- Take-home programming and verification assignments: 40% of the final grade (3 or 4 labs)
- Formal verification project: 60% of the final grade (~10 weeks, in teams of 1 to 4 students)

Supervision

Office hours	Yes
Assistants	No
Forum	Yes

Resources

Moodle Link

- https://go.epfl.ch/CS-511