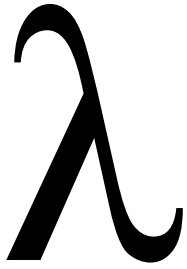
Courses we teach at RUB

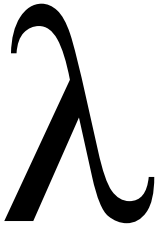


1. **Functional Programming** (SS 2024)
2. **Proofs are Programs** (WS 2024/25)
3. **Foundations of Programming Languages, Verification, and Security** (SS 2025)

Roberto Blanco, Clara Schneidewind, Cătălin Hriţcu

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

1. Functional Programming



- **Write computations as mathematical functions**
  - **using recursion, immutable datatypes, and pattern matching**
  - **limit side-effects**, such as **mutating** stateful data structures
- **Functional languages have some practical success**
  - **Meta** (OCaml, Haskell, Rust), **Microsoft** (OCaml, F#, F*, and Rust), **X** (Scala), **Mozilla** (Rust), **Google** (Rust), **Amazon** (Rust), **Financial industry**, **Blockchains**, ...
- **Not yet fully mainstream, but ...**

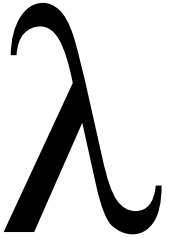
  - **Many cool ideas already adopted by mainstream languages**:
    - Lambdas, Generics in Java/C#, Rust's type system, datatypes, pattern matching (most admired language on Stack Overflow for the last 11 years!)
  - **Functional programmers often earn more** (Stack Overflow developer survey)
  - **Functional programs are concise, elegant, beautiful**
    - This makes reasoning about programs easier, both informally and formally

2. Proofs are Programs

- Follow up course **directly builds on functional programming** to **provide a gentle introduction to formal verification in Coq**
  - **Coq proof assistant is based on functional programming**
    - Coq's dependent types more powerful than OCaml types
    - Can express and prove specifications for programs
  - **Coq helps build formal proofs interactively**
  - **Proving in Coq is like programming**
    - gamified, addictive, and lots of fun
    - if you like programming, you will also like Coq proofs
  - **This helps you deeply understand proofs**
  - **In fact, formal proofs are just purely functional programs**
    - Curry-Howard: deep connection between logic and functional programming

3. Foundations of ...

- **Programming Languages**
  - formalize simple imperative and functional languages in Coq
  - type systems, program transformations, simple compilers
  - semantics, metatheory (proving properties of the language)
- **Verification**
  - Hoare Logic: verify imperative programs
  - Relational Hoare Logic: program equivalence and security
- **Security**
  - Information flow control: preventing direct + indirect leaks
  - Preventing timing side channels for crypto code: cryptographic constant time, speculative constant time



Three very hands on courses

1. **Functional Programming** (SS 2024)
2. **Proofs are Programs** (WS 2024/25)
3. **Foundations of Programming Languages, Verification, and Security** (SS 2025)

- **Based on 4 book volumes for lecture notes**
- **Many exercises in OCaml and Coq**
  - Automatic grading, immediate feedback
  - Taking gamification to the next level! It's fun!
- **Better understand programming and proving!**