This course

1. Logic and proofs
2. Functional programming
3. Program verification
   - Using the Coq proof assistant
   - Curry-Howard correspondence
     - proofs = purely functional programs
     - bridge between logic and computer science
Next steps

• Software Foundations and other Coq books

• More about the Curry-Howard correspondence

• More about functional programming

• Verifying programs with side-effects in F*
Software Foundations

• Volume 1: Logical Foundations
  – More exercises: advanced, optional
  – More chapters: Regular expressions, While programs, Lexing and Parsing, More automation, Extracting ML from Coq

• Volume 2: Programming Language Foundations
• Volume 3: Verified Functional Algorithms
• Volume 4: QuickChick: Property-Based Testing in Coq

https://softwarefoundations.cis.upenn.edu
Other Coq books, more advanced

• Adam Chlipala (MIT):
  – Certified Programming with Dependent Types
  – Formal Reasoning About Programs

• Ilya Sergey (Yale-NUS College):
  – Programs and Proofs -- Mechanizing Mathematics with Dependent Types

• Assia Mahboubi and Enrico Tassi (Inria)
  – Mathematical Components book
More about the Curry-Howard correspondence

• Phil Wadler's *Propositions as Types* paper
  – various talks available online too

• Xavier Leroy's College de France course
  – Programmer = démontrer ?
  La correspondance de Curry-Howard aujourd'hui
More about functional programming

• **OCaml MOOC** -- Classes Start: 22 September 2019

• **Book**: *Real World OCaml*, Functional Programming for the Masses
Verifying programs with side-effects in F*

- Functional programming language with effects
  - like OCaml, Haskell, F#, ...

- Semi-automated verification system using SMT
  - like Dafny, FramaC, Why3, ...

- Expressive core based on dependent type theory
  - like Coq, Agda, Lean, ...

https://fstar-lang.org