Writing and Verifying Functional Programs in Coq

Cătălin Hrițcu
Inria Paris
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
Logic and proofs

• Foundation of mathematics and computer science
  – formal proofs with respect to inference rules

• This course: constructive higher-order logic
  – constructive, aka intuitionistic logic:
    • a proposition is true if one can construct a proof
    • philosophically rejects excluded middle (P ∨ ¬P, classical logic)
  – higher-order: can quantify over propositions (∀P. P), predicates (∀Q x. Q x), relations (∀R x y. R x y), ...
Logic and computer science

• Logic and CS greatly influenced on each other, e.g.:
  – automated theorem provers (e.g., SAT and SMT solvers)
  – 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 = purely functional programs
Functional programming

- **Try to write computations as pure functions**
  - *without side-effects,* such as mutating the heap
    - sorting a list in place (imperative) vs into a new list (functional)
  - **Coq is purely functional = zero side-effects**
    - all computations are mathematical functions (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 programming languages have practical success
  – Facebook (OCaml, Haskell), Docker (OCaml), Twitter (Scala)
  – Financial industry: Jane Street (OCaml), banks (Haskell, ...)
  – Blockchains: Tezos (OCaml), Cardano (Haskell, Rust), ...

• Not yet mainstream, but ...
  – Functional programmers earn more (Stack Overflow survey)
  – Many ideas already been adopted by mainstream languages:
    generics and Lambdas in Java/C#, Google's Map-Reduce, ...
  – Makes formal verification and informal reasoning easier
Formal verification in proof assistants

• **Machine-checked proofs of mathematical theorems**
  – the 4-color and Feit-Thompson theorems (Coq+SSReflect)
  – Hales' proof of Kepler conjecture (HOL Light and Isabelle)

• **Formally verified programs**
  – **Proving mathematically that a program satisfies a specification**
  – the CompCert compiler (Coq)
  – the seL4 operating system (Isabelle/HOL)
  – the Everest HTTPS stack: EverCrypt, EverParse, miTLS (F*)
  – hot topic: verification of smart contracts
This course

• Write purely functional programs in Coq
  – natural numbers, lists, regular expressions, ...

• Verify these programs by proving theorems about them
  – case analysis, induction, inversion, ...

• Curry-Howard correspondence
  – proofs = purely functional programs

• Logical Foundations -- book written entirely in Coq

• Ask questions, interact

• Exercises, materials, website