Introduction to Coq

SoC TaeYoung Kim

KAIST, 수학문제연구회

July 19, 2020

What is Coq?

Coq is

- Interactive Theorem Prover
- Formal Proof System
- Computer-aided proof

How we prove something?

1. Define properties, functions, elements
2. State proposition
3. Apply number of tactics to prove goal given hypothesis
   - rewrite equality
   - apply proved theorems
   - induction
4. Qed

Simple simulation (Proposition as type)

To Prove

True	Single element 1
False	No proof object (empty type)
$A \to B$	Function from type A to type B Given proof of A, construct proof of B
$\forall x : T, A(x)$	Function from type T to type $A(x)$ Given element x, construct proof of $A(x)$
$\exists x : T, A(x)$	Pair of element $x : T$ and type $A(x)$
$\neg A$	Function from type A to type False If some element in type A exists, no function from A to False

Difference with other mathematics

- Purely constructive
- Not set theory, but type theory.
  In fact, Set is smallest universe of type.
- Proof is also a mathematical object.
- Many proofs rely on (structural) induction, or coinduction.

Difference with other programming language

- Purely functional
- Dependent Types
  In specific, dependent product type and dependent function type.
- All functions always terminate.
  If system can't ensure this, you should prove this.
- Deterministic computation
  No random, input.

Underlying Theories

- Dependent Type Theory
  Type theory, but type may depend on argument.
  Required to define first order propositions.
- Proof Theory
  A proposition is true iff there exists proof object of that type.
- Constructive Mathematics
  We need to construct object to prove existence.
  We can't say "Suppose there is no ..."
- Curry-Howard Correspondence
  There is correspondence between logic and computation.

Applications-mathematics

Theorem (Feit-Thompson)

Every finite group of odd order is solvable.

Theorem (4-color)

Every planar graph is 4 colorable.

These two theorem's proof is formalized by Coq.

Example (MathComp)

Coq library for formalization of mathematical theory

Applications-programming

Example (CompCert)

Fully verified C-compiler

Example (Iris logic)

Logic framework for reasoning on concurrent higher order programs

Example (KAIST Concurrency and Parallelism Laboratory)

Works for designing and verifying concurrent program and system

Pros and Cons

Advantages

- Assurance for truth of proof
- Strong automation for proof
  Pattern matching hypothesis and goal, omega, ring, auto, etc.

Disadvantages

- Need to check every detail.
- Every proof is constructive, no law of excluded middle, axiom of choice.

How to study?

This is programming language, so practice is important.

- Formalizing mathematics
  - Algebra : Good
  - Topology : Hard
  - Analysis : Very hard
- Verifying algorithms (This is what I mostly do)
- Logic theory

Concrete goals

- Formalizing Modern Algebra 1
- Formalizing Number Theory
- Logical Foundation - Software Foundation Series
- Coq'Art
- MathComp tutorial

Other theorem prover

- HOL
  Used on deep learning aided proof. See HOList paper.
- Z3
  Automated theorem prover, developed by Microsoft.
- Lean, Isabelle

Example-recursive function

```
Fixpoint sum (n : nat) : nat :=
  match n with
  | 0 => 0
  | S n' => n + (sum n')
  end.
```

decreasing on 1st argument n.

$$\text{sum } n = \sum_{i=0}^{n} i$$

Example-Proposition

```
Theorem summation :
  forall n, sum n + sum n = n * (n + 1).
Proof.
  induction n.
  - simpl. reflexivity.
  - simpl. apply eq_S. rewrite <- plus_assoc.
    rewrite <- plus_assoc. apply f_equal2_plus.
    reflexivity. simpl.
    rewrite Nat.add_succ_r. apply eq_S.
    rewrite plus_comm. rewrite <- plus_assoc.
    rewrite IHn. rewrite Nat.mul_succ_r.
    rewrite plus_comm. reflexivity.
  Qed.
```

Example-automation

```
Theorem summation_ring :
  forall n, sum n + sum n = n * (n + 1).
Proof.
  induction n.
  - simpl. reflexivity.
  - simpl.
    replace (n + sum n + S (n + sum n)) with
            ((sum n + sum n) + n + S n).
    rewrite IHn. ring. ring.
    Qed.
```

How to Install

`https://coq.inria.fr/`
Current stable version is 8.11.2
Current beta version is 8.12.0
Don't forget to modify your enviroment variable 'PATH' to contain path to coq executable.

How to use

Best : Your favorite IDE + Coq plugin
Good : CoqIde
Bad : coqc
Never : Text editor

Supplementary materials

You can see functional programming related materials here. I recommend you to read 'First order function', 'Type systems'. 'Parametric Polymorphism'.
https://hjaem.info/articles/main
Coq'Art https://www.labri.fr/perso/casteran/CoqArt/
'Programs and Proofs' https://ilyasergey.net/pnp/
'Software Foundation' https://softwarefoundations.cis.upenn.edu/current/index.html
MathComp Documentation
https://math-comp.github.io/documentation.html
'Coq Workshop' https://coq-workshop.gitlab.io/2020/
'CoqPL' https://popl19.sigplan.org/track/CoqPL-2019
'ITP contest' https://competition.isabelle.systems/