

Formalizing and Proving Theorems in Coq — Lecture 1

Tobias Kappé

Housekeeping

- ▶ Best way to reach me is `t.kappe@uva.nl`.
- ▶ Website: `https://kap.pe/coq`.
- ▶ Pass/fail (attendance, homework and projects).
- ▶ 6EC course, so expect a full-time workload.
- ▶ You should have installed Coq by now.



Today's lecture

- ▶ Why should you be using a proof assistant?
- ▶ What makes a proof assistant?
- ▶ Introduction to Coq, including demonstration.
- ▶ A note on Coq's logical foundations.
- ▶ When should you *not* be using a proof assistant?

On writing proofs

- ▶ Writing proofs is hard: lots of things can go wrong.
- ▶ Reading proofs is also hard: prose, omitted steps, etc.

Claim (Archimedes)

The regular tetrahedron can tile three-dimensional space.

Opposite is true!

Conjecture (Carmichael)

For $n \in \mathbb{N}$, there exists a number $m \neq n$ such that $\phi(n) = \phi(m)$.

(Here, ϕ is Euler's totient function.)

Flawed proof!

Fool me once

So, how do we prevent mistakes?

- ▶ Have other people read your proof (always good).
- ▶ Have a computer check your proof for you (also good).

What makes a proof assistant

Absolutely necessary:

- ▶ A small but powerful *logical system*.
- ▶ Language to state *definitions* and *claims*.
- ▶ Language to precisely present *reasoning steps*.
- ▶ Support for automatically *checking proofs*.

Nice to have:

- ▶ *Stepping* through a proof's statements.
- ▶ Support for *automating* boring proofs.



Your projects

Group exercise:

- ▶ Form groups of 2–3.
 - ▶ Explain your project idea(s) to your group.
 - ▶ Discuss possible difficulties and pitfalls.
 - ▶ Report back to the group.
- } 10–15 minutes

Proof Assistants

Lots of systems around:

- ▶ Automath (De Bruijn et al., 1967)
- ▶ Mizar (Trybulec, 1973)
- ▶ LEGO (Luo & Pollack, 1992)
- ▶ LF (Harper et al., 1993)
- ▶ ALF (Magnusson & Nordström, 1994)
- ▶ Twelf (Pfenning & Schürmann, 2002)

Currently popular:

- ▶ Nuprl (Constable, 1984)
- ▶ Isabelle/HOL (Paulson, 1986)
- ▶ **Coq** (INRIA, 1989)
- ▶ Agda (Norell, 2007)
- ▶ Lean (de Moura, 2013)
(see *Logical Verification*)

NB: these lists are incomplete!

The Coq proof assistant

- ▶ Built on *Calculus of Inductive Constructions*.
- ▶ Contains the *Gallina* programming language.
- ▶ Proof scripting using the L_{tac} language.
- ▶ Several widely acclaimed applications:
 - ▶ The Four-Color Theorem (Gonthier et al., 2005).
 - ▶ Feit–Thompson theorem (Gonthier et al., 2012).
 - ▶ CompCert, a verified C compiler (Leroy, 2006).



Typical Coq workflow

1. Encode your claim in Gallina (more on this later).
2. Enter *proof mode* by writing `Proof`.
3. Work *backwards* from the goal by applying *tactics*.
4. Repeat the last step until there are *no more goals*.
5. Close the proof with `Qed`.



Demo

Tactics seen so far

1. `intro`: transforms a goal $P \rightarrow Q$ into Q , adds hypothesis $H: P$.
2. `destruct H`: can do more than one thing:
 - ▶ break $H: P \wedge Q$ into hypotheses $H: P$ and $H': Q$.
 - ▶ do a case distinction on $H: P \vee Q$, yielding goals with $H: P$ and $H: Q$
3. `split`: can do more than one thing:
 - ▶ split a goal $P \wedge Q$ into *two* goals P and Q .
 - ▶ split a goal $P \leftrightarrow Q$ into goals $P \rightarrow Q$ and $Q \rightarrow P$
4. `exact H`: discharges goal P given that $H: P$ holds.
5. “bullet points” (`-`, `+`, `*`, `--` etc) zoom in a subgoal.
6. `left/right`: turn goal $P \vee Q$ into P or Q respectively.

Tactics seen so far (continued)

7. `apply` can be used in multiple ways:

- ▶ `apply H` turns a goal Q into a goal P , given $H: P \rightarrow Q$.
- ▶ `apply H in H'` turns hypothesis $H': P$ into $H': Q$ given $H: P \rightarrow Q$.

8. `rewrite` can be used generally, or specifically:

- ▶ `rewrite H` substitutes P for Q in the goal, given $H: P \leftrightarrow Q$
- ▶ `rewrite H with ...` forces an instantiation of the parameters of H .

Working with negation

- ▶ In Coq, negation of P is denoted by $\sim P$.
- ▶ But really, $\sim P$ is notation for $P \rightarrow \text{False}$.
- ▶ This means that we can intro and apply negations!

Logical foundations



Coq's logic is intuitionistic! By default, there is no

- ▶ *law of excluded middle* — $P \vee \sim P$
- ▶ *negation elimination* — $\sim\sim P \rightarrow P$
- ▶ *DeMorgan's second law* — $\sim(P \wedge Q) \leftrightarrow \sim P \vee \sim Q$

See the homework for more.



When should you *not* use a proof assistant?

- ▶ Spelling out every little detail can be tedious.
- ▶ Not always a great way to *discover* a proof.
- ▶ You need to recognize when to extend the underlying logic.
- ▶ Not completely understanding the language can be a pitfall.

Next lecture

- ▶ Inductive datatypes and predicates.
- ▶ Recursive functions.
- ▶ Proof by induction.
- ▶ First-order reasoning.

Homework:

- ▶ Complete the missing proofs in `lecture-1.v`.