Moderator: Bas Spitters

Andrej Bauer Cyril Cohen Robbert Krebbers
Guillaume Melquiond Anders Mortberg Karl Palmskog

Panel on libraries

The Coq Workshop 2020
online

HoTT & the Future of Formalization

Panel discussion

Andrej Bauer (University of Ljubljana) – July 5, 2020

The HoTT library

- I started it because I did not understand what Vladimir Voevodsky was doing in his Foundations library.
- I learned HoTT through formalization in Coq.
- The library would not exist without generous help from Hugo Herbelin, Bruno Barras, Assia Mahboubi, Cyril Cohen, and others.
- Support from Coq developers was essential.
- It has since grown beyond any expectations.

The future?

- Encourage young people to formalize mathematics
- Do not assume 20th century formalisms are suitable for formalized mathematics
- Educate mathematicians
- Build more & better tools
- Do not try to build the ultimate library.
- Do not worry too much about interoperability.

The Mathematical Components Library

A short retrospective and design principles

Presented by Cyril Cohen,
for the Mathematical Components developers

A short retrospective of the core library

- 2005: Creation by Gonthier & Werner for proving The Four Color Theorem
- 2006: **The Mathematical Components team**, with support from MSR-Inria joint center, settles to prove the Odd Order Theorem (Feit Thompson)
- 2006-12: USB drive → first svn commit on gforge.inria.fr
- 2008-04: First public release named *ssreflect-1.1*
- 2012-09: Completed The Odd Order Theorem and release of *ssreflect-1.4*
- 2014/2015: Switch to GitHub.com and separation between
  - *ssreflect-1.5* The Small Scale Reflection tactic language
  - ***mathcomp-1.5*** **The Mathematical Components Library**
- 2017-10: The SSReflect tactic language is included in *Coq 8.7.0*
- 2020-06: Latest release of *mathcomp-1.11.0*

Maintenance, design and engineering principles

- Compatibility over several Coq versions (8.7 → 8.12, for mathcomp 1.11.0)
- Mathematical Structures encoded by Packed Classes in Canonical Structures
- Only SSReflect + limited Small Scale Automation
- Policy on proof scripts:
  - Variables are always named explicitly, in introductions and generalizations (case, elim)
  - 1 line (≤ 80 char) = 1 reasoning step
  - 1 terminator (by, done, exact) = 1 closed subgoal
  - Rewritten frequently to use and test new features and styles
  - Interleave *readable forward steps* with *compact procedural paragraphs*
  - Goal: be maintainable (easy to repair)
- A focus on reasonably complete API (theories) and naming conventions
- No axioms in the *main core repository*, “classical reasoning” is encapsulated by boolean predicates, `eqType` and `choiceType`.

Many related libraries and projects

- The Four Color Theorem (ported to “modern mathcomp” on 2019-04-25)
- The Odd Order Theorem (distributed separately from mathcomp library)
- Apery’s proof of irrationality of ζ(3)
- Shannon’s information theory
- Solutions to the POPLmark Challenge
- Mathcomp-Analysis: Classical analysis compatible with mathcomp
- Partial Commutative Monoids Library (FSCL-PCM)
- Various extensions (finite maps, elliptic curves, polyhedra, graphs, …)
- Various theorems (Sums of squares, QE on RCF, Grobner, Lindemann, ...)
- *… and many more results in various domains (Real algebraic geometry, Graph theory, Homology, Concurrency, Robotics, Modal logic, etc)*

https://math-comp.github.io/papers.html

About me

Robbert Krebbers (TU Delft, The Netherlands)

- Active Coq user since 2010
- Mechanized efficient reals using the math-classes and CoRN libraries (2010)
- PhD on mechanizing C (2011–2015)
- Lead-developer (with Ralf Jung and Jacques-Henri Jourdan) of the std++ and Iris libraries (2015–now)
- Nearly all my papers are mechanized in Coq

std++ "extended standard library"

- Focused on mechanization of PL research
- Large collection of definitions and lemmas for lists, sets, multisets, maps
- Type classes for notation overloading ($\varnothing$, $\cup$, do notation, ...)
- Type classes for properties like decidable equality, countability, finiteness, ...
- Tactics for automation (`set_solver`, `naive_solver`, ...)
- Axiom-free and dependency-free
- Uses setoids, *but* as little as possible
- Developed during my PhD (2011-2015)
- Now part of the Iris project with many external contributions

Iris "framework for concurrent separation logic"

- Comes with a tactic language for separation logic proofs (IPM/MoSeL)
- Highly extensible and parametrized
- Used in ca. 30 publications to prove a variety of properties (safety, refinement, security, ...) of a variety of languages (ML-like, Rust, Scala, C, ...)
- Uses type classes and canonical structures for extensibility
- Uses ssreflect (mostly the `rewrite` tactic) and std++
- Developed by Ralf Jung, Jacques-Henri Jourdan, and me, with many external contributors

Reflection on developing Coq libraries

Awesome things

- ☑ The stability and quality of Coq releases is great
- ☑ Coq is amazingly extensible (Iris would not be possible without that!)

Reflection on developing Coq libraries

Awesome things

- ✓ The stability and quality of Coq releases is great
- ✓ Coq is amazingly extensible (Iris would not be possible without that!)

Things that need improvement

- ✗ Unification is unreliable (according to some Coq devs `apply` is obsolete ☹)
- ✗ `simpl`/`cbn` are broken (a well-behaved simplification mechanism is crucial)
- ✗ Type classes v.s. canonical structures (both have their issues)
- ✗ Ltac (give me data types, opt-in instead of opt-out backtracking, exceptions, ...)
- ✗ Too many data types for the same thing (take the number types for example)

Guillaume Melquiond

Maintained libraries

- Flocq: formalization of floating-/fixed-point arithmetic.
- Coquelicot: formalization of classical real analysis.
- Gappa: automation for floating-point arithmetic proofs.
- CoqInterval: automation for real analysis proofs.
- Why3: consistency of Why3's standard library.

Features

- About 200k lines of Coq.
- Backward compatibility as far back as 8.6–8.8.
- Licensed under LGPL or equivalent.
- Packaged using Opam.

Anders Mörtberg

About me:

- Currently assistant professor in mathematics at Stockholm University
- Started working with both Agda and Coq around 2010
- Phd: developed CoqEAL library and formalized constructive algebra using SSReflect/MathComp
- Postdoc: made substantial contributions to the UniMath library
- I've also developed multiple experimental proof assistants and typecheckers for cubical type theories (cubical, cubicaltt, yacctt...)

Current work

These days I'm mainly working on Cubical Agda—a fully fledged dependently typed programming language for cubical type theory

Since 2018-10-15 I've been maintaining and developing a library with Andrea Vezzosi called `agda/cubical` (by now 41 contributors, > 31k LOC, 300 files):

`https://github.com/agda/cubical/`

Question: will there be a cubical mode for Coq?

Proof Engineering for Libraries

Quality of Libraries

- mutation analysis can find underspecified definitions
- `EngineeringSoftware/mcoq`

Maintenance of Libraries

- scripts/templates for automation can assist maintainers
- `coq-community/templates`

Coding Conventions

- use tools to suggest lemma names and spacing
- `EngineeringSoftware/roosterize`

Regression Proving

- Avoid reproving every proof in every commit!
- `palmskog/chip`

`https://setoid.com` - `https://proofengineering.org`