COQ DEVELOPMENT TEAM SESSION

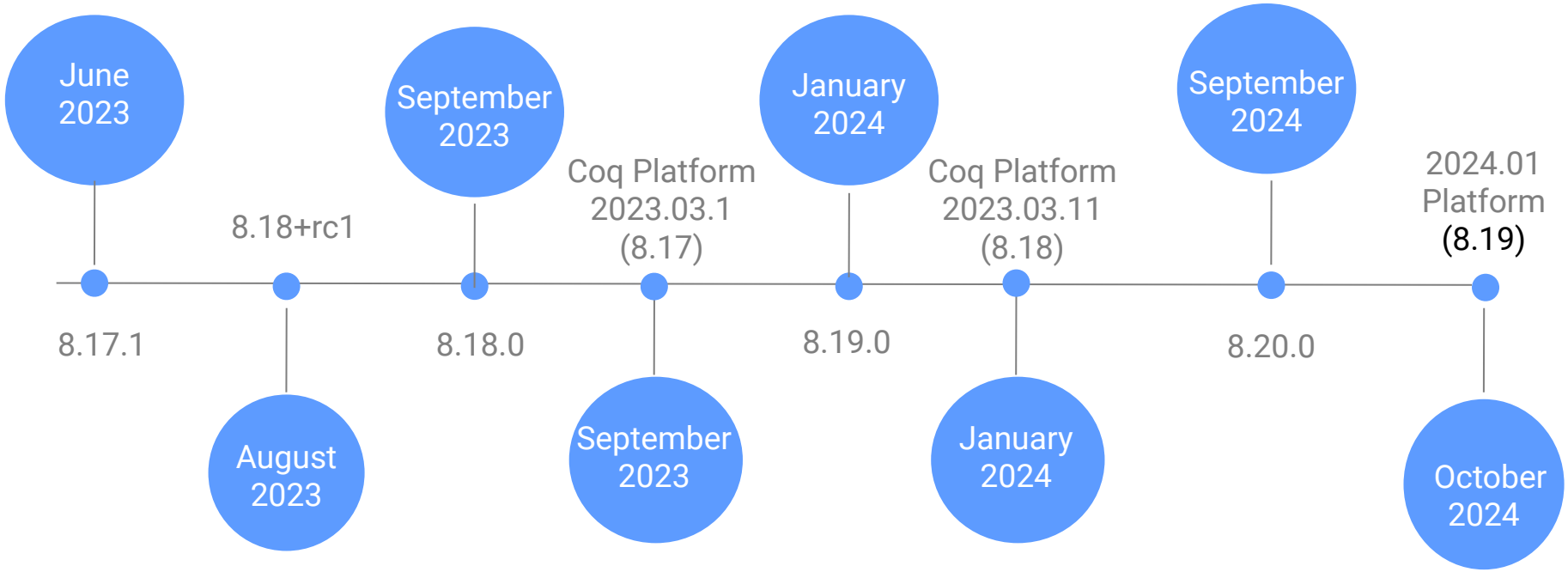
Matthieu Sozeau &
Coq core team

Coq Workshop 2024
September 14th 2024

OUTLINE

1. Coq 8.18, 8.19 & 8.20
2. Coq Platform Updates
3. Coq Future & Roadmaps
4. Highlights
5. Q & A

Coq and Platform Release Timeline



Coq 8.18-8.19 Features

https://coq.inria.fr/refman/changes.html

- Notations activation/deactivation
- Temporary scopes, multiple scopes for Arguments
- Ltac2 improvements: richer APIs, case compilation, bugfixes
- Sort polymorphism and unification of sorts.

  Generic definitions over Prop, SProp and Type.
- Stdlib improvements: arithmetic libraries, lists, analysis

Version 8.20

Summary of changes

Coq version 8.20 adds a new rewrite rule mechanism along with a few new features, a host of improvements to the virtual machine, the notation system, Ltac2 and the standard library.

We highlight some of the most impactful changes here:

- User-defined rewrite rules
- primitive strings
- A lot of work went into reducing the size of the bytecode segment, which in turn means that .vo files might now be considerably smaller.
- A new version of the docker-keeper compiler to build and maintain Docker images of Coq.

Coq 8.18-8.19 Changes

https://coq.inria.fr/refman/changes.html

- Default localities for hints and instances
- Improved control over warnings, providing better support for deprecations. Library (i.e. file) deprecation available.
- Lazy, simpl, cbn and eval now can do head reduction
- Precise profiling support
- OCaml 5 compatibility (perf caveats, no `native_compute`)

8.20 Breaking Changes

Notable breaking changes:

- Syntactic global references passed through the `using` clauses of **`auto`**-like tactics are now handled as plain references rather than interpreted terms. In particular, their typeclass arguments will not be inferred. In general, the previous behaviour can be emulated by replacing `auto using foo` with `pose proof foo; auto`.
- Argument order for the Ltac2 combinators `List.fold_left2` and `List.fold_right2` changed to be the same as in OCaml.
- **`Import`**ing a module containing a mutable Ltac2 definition does not undo its mutations. Replace `Ltac2 mutable foo := some_expr.` with `Ltac2 mutable foo := some_expr.` `Ltac2 Set foo := some_expr.` to recover the previous behaviour.
- Some renaming in the standard library. Deprecations are provided for a smooth transition.

Demo

- Sort polymorphism
- Notation activation, selective imports
- Ltac2
- Rewrite Rules

OUTLINE

1. Coq 8.18, 8.19 & 8.20
2. **Coq Platform Updates**
3. Coq Future & Roadmap(s)
4. Highlights
5. Q & A

Coq Platform

A coherent *distribution* of Coq with Coq packages.

Main objectives: easy, standard, platform-independent and tested

- Installers for Windows and OS X, scripts for many Linux distros.
- Customizable! Just choose a package list (e.g. for lectures)

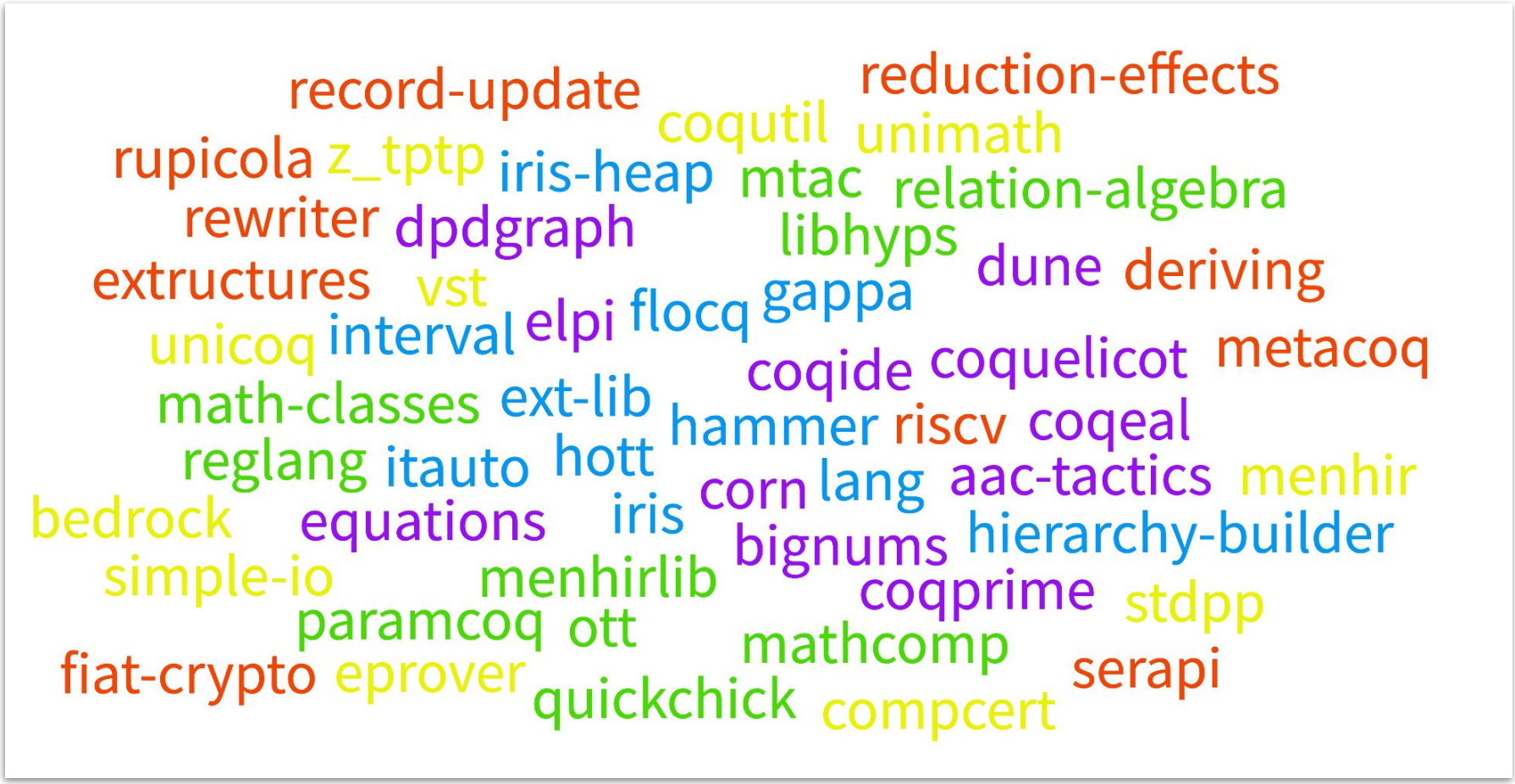
Coq Platform Charter

Maintenance: Michael Soegtrop & Romain Tetley

Editorial board: Reynald Affeldt, Andrew Appel, Yves Bertot,

Michael Soegtrop & Matthieu Sozeau

Coq Platform Packages



Coq Platform 2024.1 (8.19)

Delayed due to considerable difficulties making the build work on Windows and (Apple Silicon) Mac OS X. Snap no longer supported.

- Support by Inria Engineer Romain Tetley (VSCoq 2 maintainer)
- We are looking at hiring an engineer specifically on this topic, relieving both Romain and Michael. Position open for a 3 year term at Inria Sophia-Antipolis, that we hope to turn into a permanent position.

OUTLINE

1. Coq 8.18, 8.19 & 8.20
2. Coq Platform Updates
3. **Coq Future & Roadmaps**
4. Highlights
5. Q & A

In the pipeline

- Sort polymorphism use in the stdlib, subsuming template-polymorphism (P.M. Pédrot, G. Gilbert)
- Algebraic universes for all (PR#16022, M. Sozeau, M. Bezem)
- Ltac2 maturation as a replacement for Ltac1 (P.M. Pédrot, G. Gilbert)
- From Coq -> From Stdlib, moving Stdlib from the main repo (P. Roux)

See the short-term roadmap for Coq (CEP#69)

Renaming

The Coq Proof Assistant will be renamed The Rocq Prover (abbreviated, Rocq).

- User-survey results: split on the renaming, majority for renaming or neutral Development team majority for the renaming.
- Pronounced /ɹɑk/. Pays homage to Rocquencourt (birthplace of Coq), suggestive of rock-solid software.
- Plan: make the renaming effective this year, with an updated visual identity, website and first release. In the meantime, keep using Coq!
- We are aware of the Roc programming language and the converse is true as well. We believe it will be easy to distinguish them: they have very different use cases, and one can use Rocq Prover/ITP in case of ambiguity.

Long-term vision

https://rocq-prover.org/roadmap.html

Planned Events

- CoqPL 2025, collocated with POPL'25 in January

OUTLINE

1. Coq 8.18, 8.19 & 8.20
2. Coq Platform Updates
3. Coq Roadmap & Future
4. **Highlights**
5. Q & A

Highlight: Coq-Elpi & applications

Coq-Elpi: high-level rule-based extension language (E.Tassi)

Hierarchy-Builder: structure hierarchies à la Mathematical Components from declarative specifications (C.Cohen, K.Sakaguchi, E.Tassi)

Trakt/**Trocq**: extensible framework for transfer between theories (E.Crance, C.Cohen, A.Mahboubi)

Derive: automatic synthesis of code, eg. parametricity, equality tests, lenses, … (B.Gregoire, JC.Lechenet, E.Tassi)

Experiments around **elaboration**: type-class, canonical-structure and coercion inference via Elpi programs (D.Fissore, Q.Vermande, P.Roux)

Highlight: MetaCoq, CertiCoq, ConCert

Verified erasing compilation pipelines from **Gallina** through **λ□** to:

★ **C** compilable by CompCert/clang/gcc

CertiCoq: bootstrappable, with a verified GC! (Appel et al)

VeriFFI: link with VST (Korkut, Stark & Appel)

★ **WebAssembly:** alternative CertiCoq code generator

Certicoq-wasm (Meier et al)

★ **Malfunction** / OCaml: with a restricted, safe .mli interface

coq-verified-extraction (Forster, Sozeau & Tabareau)

★ **Web** / Smart Contract Languages (Liquidity, Elm/MidLang)

ConCert (Annenkov et al). Uses a *type erasure* phase

Highlight: Projects around Coq/Rocq

- ★ **Liber Abaci** (2022-2027): teaching mathematics using Coq. Inria project led by Yves Bertot.
- ★ **ERC Fresco** (2021-2026), **Fast and Reliable Symbolic Computation**, Assia Mahboubi (PI), Guillaume Melquiond
- ★ LLM4Code: Inria "Défi" (2024-2029), Guillaume Baudart and Marc Lelargue

Flèche / coq-lsp / jsCoq Update

- **Flèche:** New Document and Theory Manager for Coq; *replaces* SerAPI
  core component for jsCoq 2.0, fcc, coq-lsp, petanque
- **Flèche Roadmap:**
  - **0.1.x** (completed)
    incremental real-time **document** checking and editing
    (E.J. Gallego Arias, G. Gilbert, G. Munch-Maccagnoni)
  - **0.2.x:** (in progress) [0.2.0 release: Aug 29th 2024]
    incremental real-time **theory** checking and editing
    (E.J. Gallego Arias, B. Shah)
- **petanque:** agent for Reinforcement Learning and proof search applications
  (E.J. Gallego Arias, G. Baudart, M. Lelarge, A. Sanchez-Stern, L. Teodorescu)
- **jsCoq 2.0:** scheduled for fall, significant improvements
  (E.J. Gallego Arias, H. Herbelin, H. Heuzard, S. Itzhaky)

Stay tuned for papers / events in the fall

Q & A Time!