Rocq Development Team Session

https://rocq-prover.zulipchat.com/#narrow/channel/558403-RocqPL'26

RocqPL 2026, Rennes, France — January 17th 2026

Matthieu Sozeau, Gaëtan Gilbert & Yann Leray
Inria & LS2N, University of Nantes, France

Outline

- 2025 News
- 2026 Teaser
- Rocq's Future

Rocq Development Team Session

https://rocq-prover.zulipchat.com/#narrow/channel/558403-RocqPL'26

2025 News

The Rocq Prover - 9.0 (March 2025)

RMs: Pierre-Marie Pédrot (& Matthieu Sozeau)

- Renaming, new identity and website
- Splitting the `Coq` Stdlib into a Rocq `Corelib` and Rocq `Stdlib` (wrapping `Corelib`)
- A single `rocq` binary dispatching to various commands

Full Changelog

The Rocq Prover - 9.0.1 (September 2025)

Fixes for inconsistencies derivable from:

- The guard checker: propositional extensionality strikes back! Correctness **and** completeness issues.
- The infamous “delta-resolver” of the module system: name equivalence quotient not being respected by substitution of functor delta resolvers during strengthening.

Full Changelog

The Rocq Prover - 9.1 (September 2025)

RMs: Gaëtan Gilbert (Pierre-Marie Pédrot)

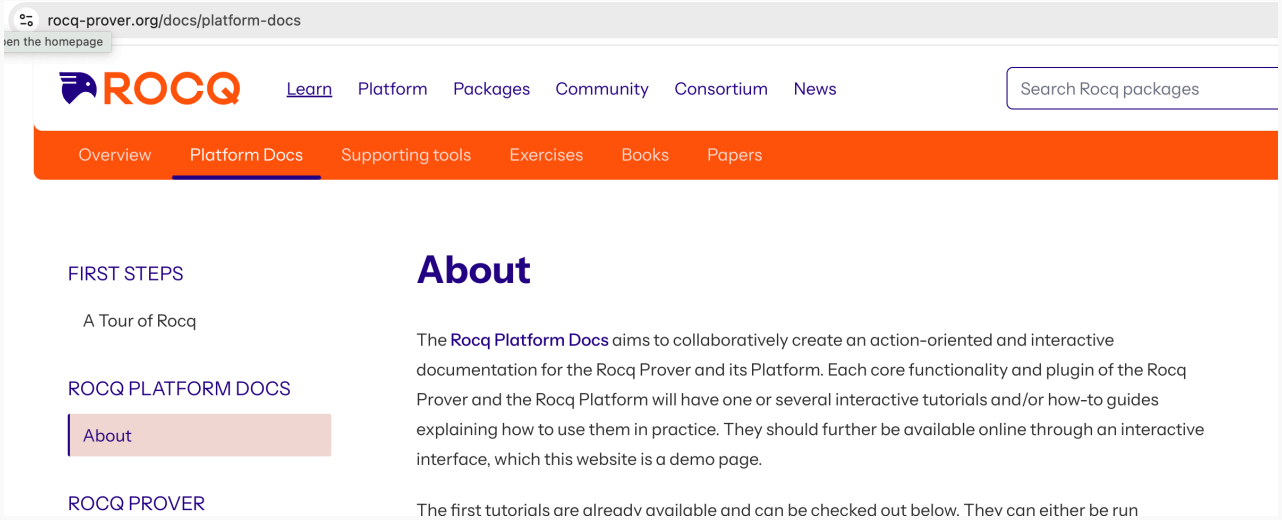
- fixed incorrect guard checking leading to inconsistencies (multiple PRs)
- sort polymorphic universe instances should now be written as `@{s ; u}` instead of `@{s | u}`
- fixed handling of notation variables for ltac2 in notations (i.e. `Notation "'foo' x" := ltac2:(...)`)
- Support for **`refine`** attribute in **`Definition`**
- Rocq can be compile-time configured to be relocatable
- extraction handles sort polymorphic definitions

Full Changelog

New Ressources

Docker-based CI/CD for Rocq/OCaml projects, Éric Martin-Dorel, Oct 2025
https://arxiv.org/pdf/2510.19089

Platform documentation, welcoming contributions!



New books

- *Security Foundations, (Vol 7 of Software Foundations)*, Cătălin Hriţcu, Yonghyun Kim and contributors

- *Modeling and Proving in Computational Type Theory*, Gert Smolka (in construction)

Social

👋 `coq-club@inria.fr` is **archived**

Rocq Zulip: https://rocq-prover.zulipchat.com

Rocq Discourse: https://discourse.rocq-prover.org/

Rocq Mastodon: `@RocqProver`

⇒ Setup your email notifications!

Rocq Development Teams

- **Rocq Core**: 11 researchers/engineers led by a coordinator: Huet 1984-1996, Paulin 1996-2006, Herbelin 2006-2016, Sozeau 2016-2026
- **Rocq Maintainers** (of the Rocq Prover projects): ~50 researchers/engineers
- **Rocq Consortium**: backed by Inria and contributors, supports 2 engineers: G. Gilbert (Core) and S. Borgogno (Rocq Platform, Docker-Rocq)
- **Rocq Community**: open source community effort of package maintance

2026 Teaser

The Rocq Prover - 9.2 (ETA March 2026)

- `Scheme` tables for robust scheme/derived principle lookups (deprecating fragile lookups by name)
  - Eliminators for nested inductive types, done right (Thomas Lamiaux @ 2.45pm), PR #21356)

  Feedback Zulip channel

The Rocq Prover - 9.2 (ETA March 2026)

- `Scheme` tables for robust scheme/derived principle lookups (deprecating fragile lookups by name)
  - Eliminators for nested inductive types, done right (Thomas Lamiaux @ 2.45pm), PR #21356)

  Feedback Zulip channel
- Support for sort elimination constraints (Rosain et al, POPL'26)

The Rocq Prover - 9.2 (ETA March 2026)

- `Scheme` tables for robust scheme/derived principle lookups (deprecating fragile lookups by name)
  - Eliminators for nested inductive types, done right (Thomas Lamiaux @ 2.45pm), PR #21356)

  Feedback Zulip channel
- Support for sort elimination constraints (Rosain et al, POPL'26)
- Rewriting with arbitrary relations supporting the Leibniz principle (in any appropriate sort).

The Rocq Prover - 9.2 (ETA March 2026)

- Automatic goal names for `eapply`,`induction`,`destruct` (Halilovic, PR #20813).

The Rocq Prover - 9.2 (ETA March 2026)

- Automatic goal names for `eapply`,`induction`,`destruct` (Halilovic, PR #20813).
- Clean support for `abstract` / environment extension in proofs / obligations

The Rocq Prover - 9.2 (ETA March 2026)

- Automatic goal names for `eapply`,`induction`,`destruct` (Halilovic, PR #20813).
- Clean support for `abstract` / environment extension in proofs / obligations
- Extended Ltac2 APIs & notation support, custom grammar entries

The Rocq Prover - 9.2 (ETA March 2026)

- Automatic goal names for `eapply`,`induction`,`destruct` (Halilovic, PR #20813).
- Clean support for `abstract` / environment extension in proofs / obligations
- Extended Ltac2 APIs & notation support, custom grammar entries
- Performance improvements with OCaml 5, on par or improving w.r.t 4.14 in some cases now

Rocq Ecosystem News

- elpi: Determinacy analysis (Fissore & Tassi), parametricity (Cyril Cohen & Vojtěch Štěpančík @ 2.15pm)

Rocq Ecosystem News

- elpi: Determinacy analysis (Fissore & Tassi), parametricity (Cyril Cohen & Vojtěch Štěpančík @ 2.15pm)
- Fresco ERC projects, Trocq (Tomas Vallejos Parada @ 3pm) & Capla: beyond Rocq as a calculator, Rocq as a CAS (Mahboubi, Melquiond, ...).
  - WIP vm_compute binding to GMP, FFIs (Borthelle)

Events

- Iris Workshop, Klosterneuburg/Vienna, Austria, June 8th-10 2026
- Formal Mathematics and Proof System Interoperability, Institut Pascal, Orsay, June 22nd - July 3rd 2026
- Rocq'n'share, EPFL, Switzerland, June 29th - July 3rd 2026
- Rocqshop, collocated with FLoC, Lisbon, Portugal, July 18th - 25th 2026
- Fresco Summer School: Machine-Checked Mathematics: an introduction to formally verified mathematics
  Oaxaca, Mexico, July 20th-31rd 2026

Rocq's Future

Current R&D: Immediate pipeline

- **Sort Polymorphism with Elimination Constraints** (Rosain et al, POPL'26)
  Generic eliminators, generic theories working over `Prop/SProp/Type/ ..`
- **Universe Level Polymorphism** (Sozeau and Bezem, RocqPL'26 @ 2.30pm)
  Algebraic universes everywhere based to a new, verified, constraint satisfaction and validity algorithms for the $(\mathbb{N}, \max, \leq)$ semilattice

Current R&D: Rewrite Rules

Extend definitional equality with decidable theories locally and modularly.

Encode the Cake and Eat it Too, Leray and Winterhalter, POPL'26.

- Work up-to associative and neutral operations, e.g., indexing by numbers, with less dependent type hassles.
- Abstraction over encodings (e.g. σ-calculus, Equations's DPM encoding, ...)
- Axiomatic quotients with a computation rule (~ Lean's theory)

Current R&D: Custom sorts

Observational Equality Meets CIC

Pujet, Leray and Tabareau, TOPLAS'25

Leibniz equality (in **`Prop`**) ⇒ Observational Equality (in **`SProp`**)

- Definitionally Proof-Irrelevant Equality: UIP is built-in
- Functional and propositional extensionality hold by definition
- Quotients by propositional relations without breaking the metatheory

OTT provides axiomatic freedom: extension with consistent axioms in **`SProp`** is safe (≠ Homotopy Type Theory).

For example, canonicity for ℕ still holds even in presence of classical axioms like excluded middle or choice.

Current R&D: Custom sorts

Ghost Type Theory / Ghost Sort

Winterhalter, ICFP'24 / Rosain, Sozeau and Winterhalter, TYPES'25

A sort for proof-relevant **and** computationally irrelevant data (~ "ghost" variables used in program logics).

⇒ Cannot influence computational content (in `Type`)

⇒ Erased at extraction time (or even conversion time)

Inductive Families Need Not Store Their Indices, Brady, McBride and McKinna

The Implicit Calculus of Constructions, Barras & Bernardo

Current R&D: Reduction and conversion

- *A Lazy, Concurrent Convertibility Checker*, Nathanëlle Courant and Xavier Leroy, POPL'26

  "Optimal" conversion with sharing, looking for a candidate to help Nathanaëlle implement it in Rocq.

- Programmable reduction and certificates for conversion, Quentin Corradi and Clément Pit-Claudel.
  - Fine-grain subterm selection and reduction strategies (teaching, term chirurgy)
  - https://systemf.epfl.ch/blog/rocq-conversion-algorithm/ Avoid dreaded non-terminating `Qed`s due to different choices being made in tactics and kernel type-checking!

Current R&D: Rocqola, Metaprogramming and Verification

- **Rocqola**: OCaml for Rocq, French ANR Project: Gabriel Sherer, Pierre-Marie Pédrot, ...
- Towards a verifiable meta-programming environment in Rocq (Mathis Bouverot-Dupuis, Yannick Forster & François Pottier)
- MetaRocq formalizations of Sort Poly with Elim Constraints, Guard Condition (Johann Rosain, Thomas Lamiaux, Forster, Sozeau, Tabareau, ...)

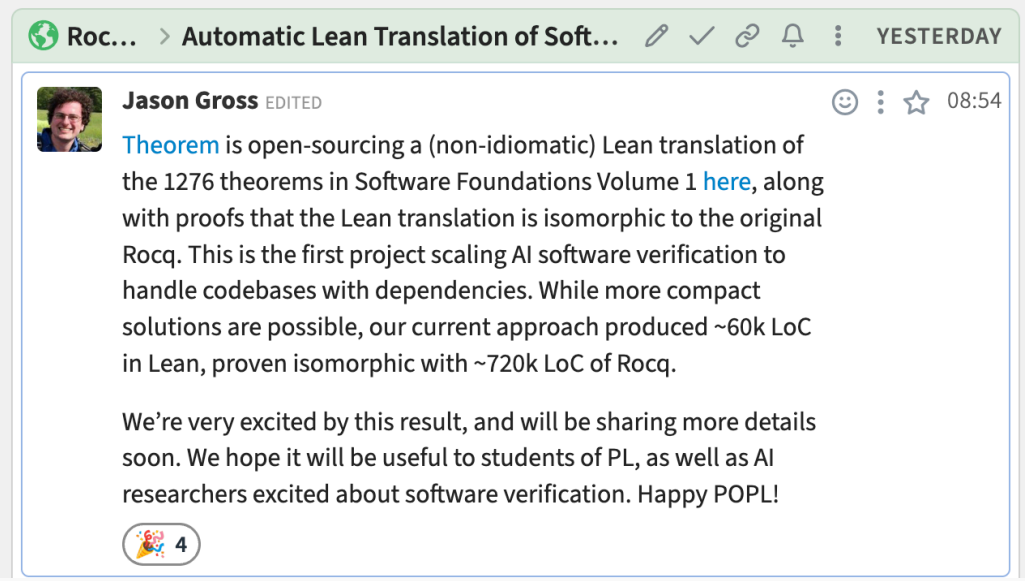
Current R&D: (Verified) Extraction

- CertiCoq: verified extraction to C/Wasm/OCaml(Malfunction).

  - CertiCoq-Wasm (Meier et al, CPP'25)
  - Verified separate compilation to Malfunction (Forster, Sozeau and Tabareau, PLDI'24)
  - VeriFFI: Verified FFI to C/VST (Korkut, Stark and Appel, POPL'25)
- Peregrine (https://peregrine-project.github.io)
  Inria, Aarhus University, IOG, Ethereum

Verified extraction middle-end ($\lambda_{\square}$)
Rocq / Agda (agda2$\lambda_{\square}$) / Lean $\Rightarrow$ Clight / Wasm / Rust / CakeML / ...

2 Elephants in The Room

- LLM4Code @ Inria
- …
- 

Rocq Development Team Session

https://rocq-prover.zulipchat.com/#narrow/channel/558403-RocqPL'26

Q&A!

RocqPL 2026, Rennes, France — January 17th 2026

Matthieu Sozeau, Gaëtan Gilbert & Yann Leray
Inria & LS2N, University of Nantes, France

Current R&D: Rewrite Rules

⇒ More comfortable deep-shallow embeddings.

Example Decidable substitution calculus on AST definitions

```
Module Type Renaming.

  Parameter renaming : Type.

  Symbol ren_var : renaming → nat → nat.
  Symbol up_ren : renaming → renaming.
  Symbol shift_ren : renaming.

  Symbol comp_ren : renaming → renaming → renaming.
  Symbol id_ren : renaming.

  Rewrite Rules ren_rew :=
  | ren_var (up_ren ?σ) 0 => 0
  | ren_var (up_ren ?σ) (S ?n) => S (ren_var ?σ ?n)
  | ren_var shift_ren ?n => S ?n
  | ren_var (comp_ren ?σ ?τ) ?n => ren_var ?σ (ren_var ?τ ?n)
  | ren_var id_ren ?n => ?n
  | comp_ren id_ren ?σ => ?σ
  | comp_ren ?σ id_ren => ?σ
  | comp_ren (comp_ren ?σ ?σ') ?σ'' => comp_ren ?σ (comp_ren ?σ' ?σ'').

End Renaming.
```

```
Definition rename (r : renaming) (t : term) : term :=
  ..

Goal forall (t : term) r, comp_ren (shift_ren r) id_ren = r.
Proof. reflexivity. Qed.
```

Ghost Sort Example: Indexed Types I

Wasteful representation

```
(* Total lookup function *)

Inductive fin : nat → Set :=
| fin0 {n} : fin (S n)
| finS {n} : fin n → fin (S n).

Fixpoint lookup {A} (l : list A) :
  fin (length l) → A :=
  match l return fin (length l) → A with
  | nil ⇒ fun (f : fin 0) ⇒
      False_rect (match f in (fin 0) with end)
  | cons a l ⇒ fun f : fin (S (length l)) ⇒
    match f with
    | fin0 ⇒ a
    | finS f' ⇒ lookup l f'
    end
  end.
```

```
(* Extraction to OCaml: *)

type fin =
| Fin0 of nat
| FinS of nat * fin

let rec lookup l f =
  match l with
  | Nil → assert false
  | Cons (a, l0) →
    match f with
    | Fin0 _ → a
    | FinS (_, f') → lookup l0 f'
```

Ghost Sort Example: Indexed Types II

Optimal representation

```
(* Total lookup function *)

Inductive fin : nat@{Ghost; 0} → Set :=
| finO {n} : fin (S n)
| finS {n} : fin n → fin (S n).

Fixpoint lookup {A} (l : list A)
  : fin (length@{Ghost; 0} l) → A :=
  match l return fin (length@{Ghost} l) → A with
  | nil ⇒ fun (f : fin 0) ⇒
      False_rect (match f in (fin 0) with end)
  | cons a l ⇒ fun f : fin (S (length l)) ⇒
    match f with
    | finO ⇒ a
    | finS f' ⇒ lookup l f'
    end
  end.
```

```
(* Extraction to OCaml: *)

type fin = nat =
| Fin0
| FinS of fin

let rec lookup l f =
  match l with
  | Nil → assert false
  | Cons (a, l0) →
     match f with
     | Fin0 → a
     | FinS f' → lookup l0 f'
```