# Canonical for Rocq

Supervisors: Pierre Boutry, Chase Norman, Loïc Pujet

## Background

### Canonical

Proving a theorem in the Rocq Prover [9] boils down to encoding the theorem as a type, and then finding a proof term that inhabits it. Ideally, we would like to have powerful tactics to help us find these proof terms, but since type theory can be used to encode arbitrary mathematical theorems, determining whether a type is inhabited is undecidable. Thus, in practice, most tactics solve a more restricted class of problems, which can be phrased in a decidable and well-understood theory (for instance, `lia` is a solver for Presburger arithmetic).

However, some tactics take a more general approach, by encoding the goal in higher-order logic (HOL), invoking solvers such as Vampire [4], and reconstructing a proof term to close the goal. These are known as *hammers* [3]. By design, these systems can only use the logical connectives of HOL, cannot directly reason with dependent types, and cannot synthesize functions or objects constructively. Additionally, hammers do not present an axiomatic proof term to the user, as such a term would be much too large to be read.

In a recent contribution [6], Norman and Avigad introduced Canonical, the first solver for type inhabitation in dependent type theory. Canonical searches exhaustively for terms of any type involving dependent products (Π-types), record types, inductive types, and let definitions.

### From an inhabitation solver to a Lean tactic

Canonical does not work with the type theory of any particular proof assistant – instead, it works with its own type theory. This theory is rather barebones by default, but it can be extended with constants and equations to simulate more sophisticated type theories such as the ones used by Agda [1], Lean [5] and Rocq.

In order to use Canonical in Lean, Norman and Avigad have designed a tactic which encodes the current goal in the type theory of Canonical, calls the solver on it, and finally reconstructs a Lean proof term that the user can paste in their proof script. This tactic manages to solve 84% of the exercices in the *Natural Number Game* [2], which is perhaps the most popular tutorial for Lean. Canonical can also work with higher-order objects, for instance it manages to find a proof term for Cantor's diagonal argument.

## Goals of the internship

The goal of the internship is to write a Rocq tactic that invokes canonical. The first step is to encode the theory of Rocq into the type system of Canonical, which requires the intern to get

acquainted with the internals of Rocq. The second step is to write the foreign function interface that will reify a Rocq goal and call Canonical on it. To achieve this, the intern will pick a language for metaprogramming in Rocq such as OCaml, Metarocq [7] or Rocq-Elpi [8]. Finally, the tactic will use the result found by Canonical to reconstruct a proof term in the syntax of Rocq.

## Required skills

The intern is expected to have some experience with functional programming, with dependent type theory, and with the use of at least one proof assistant based on dependent type theory.

## Organization

The internship will take place in the University of Strasbourg, where Pierre Boutry and Loïc Pujet are based. Chase Norman is based in Carnegie Mellon University, Pittsburgh (USA), and thus we will organize a weekly or bi-weekly video meeting with him.

## References


[1] Ana Bove, Peter Dybjer, and Ulf Norell. A Brief Overview of Agda – A Functional Language with Dependent Types. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics*, pages 73–78, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Kevin Buzzard, Jon Eugster, Alexander Bentkamp, Mohammad Pedramfar, and Patrick Massot. Natural number game. Available at `https://adam.math.hhu.de/#/g/leanprover-community/nng4` (2025/03).

[3] Łukasz Czajka and Cezary Kaliszyk. Hammer for Coq: Automation for dependent type theory. *Journal of automated reasoning*, 61(1):423–453, 2018.

[4] Laura Kovács and Andrei Voronkov. First-Order Theorem Proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 1–35, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, pages 625–635, Cham, 2021. Springer International Publishing.

[6] Chase Norman and Jeremy Avigad. Canonical for Automated Theorem Proving in Lean. In Yannick Forster and Chantal Keller, editors, *16th International Conference on Interactive Theorem Proving (ITP 2025)*, volume 352 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:20, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[7] Matthieu Sozeau, Abhishek Anand, Simon Boulier, Cyril Cohen, Yannick Forster, Fabian Kunze, Gregory Malecha, Nicolas Tabareau, and Théo Winterhalter. The MetaCoq Project. *Journal of Automated Reasoning*, February 2020.

[8] Enrico Tassi. Elpi: rule-based meta-language for Rocq. In *CoqPL 2025 - The Eleventh International Workshop on Coq for Programming Languages*, Denver (Colorado, USA), United States, January 2025.

[9] The Rocq Development Team. The Rocq Prover. `https://doi.org/10.5281/zenodo.15149629`, April 2025.