Automatisation, maintenance et réutilisation des démonstrations dans l'assistant de preuves Coq : applications aux mathématiques et à la géométrie

Directeur de thèse : Nicolas Magaud

January 30, 2023

1 Résumé (en français)

Les assistants de preuve comme Coq permettent, dans un même outil, d'implanter sous forme fonctionnelle des objets mathématiques et des opérations sur ces objets, ainsi que de démontrer formellement leurs propriétés. Ce travail de spécification et de preuve est confié à l'utilisateur, le système vérifiant simplement que le raisonnement construit est correct. Cette approche permet de s'affranchir des limitations des outils automatiques et de tirer profit, à plein, de l'intelligence et de l'inventivité humaines. Néanmoins, plus les preuves sont complexes, et plus il faut fournir aux développeurs des outils pratiques pour les construire, les maintenir et les réutiliser facilement.

L'objectif de cette thèse sera de rendre plus robustes et facilement réutilisables, y compris d'un système à un autre, de telles preuves. On cherchera à étendre le champ d'application d'outils de preuves automatiques comme Vampire, CVC5 ou Z3 et à faciliter leur interaction avec Coq (en transformant les traces de preuves produites en certificats vérifiables par Coq). On étudiera également comment rendre plus robustes, génériques et interopérables, les étapes de raisonnement non automatisées. Leur réutilisation pour prouver formellement de nouveaux résultats en sera ainsi facilitée. Les développements sur la géométrie réalisés dans l'équipe serviront de premiers cas d'étude et permettront d'évaluer les performances des outils proposés.

2 Sujet détaillé (en anglais)

Abstract: Proof assistants like Coq are increasingly popular to help mathematicians carry out proofs of the results they conjecture. However, formal proofs remain highly technical and are difficult to reuse. In this thesis, we propose to design and implement new tools to improve the robustness, the inter-operability and the reuse of formal proof developments. We hope this will make proving new results formally easier.

Keywords: automated theorem proving, interactive theorem proving, Coq, tactics, interoperability of systems

Expected skills : software development, interactive theorem proving, Coq functional programming, logic

Skills to be acquired during the PhD programme: building large proof developments, interoperability of proof systems, geometry and combinatorics, implementation of proof transformations and translations from one system to another.

2.1 Scientific context

Proofs assistants such as Coq [7, 4] and Lean [9] allow, using the same infrastructure, not only to implement mathematical objects and their operations using the functional programming paradigm, but also to formally prove some of their properties. This specification and proof task is devoted to the user whereas the system checks that the reasoning arguments are correct. Thanks to this approach, relying on human intelligence, there are no limitations related to the power of the automatic decision procedures available. However, as proofs grow in size and complexity, it becomes unavoidable to have practical tools to help the user carry out some proof steps automatically. Decision procedures for decidable fragments of the considered theories are sometimes available but they are a lot less powerful than automated first-order provers such as Vampire [14], CVC4 [3] or Z3 [8]. In addition to these generic provers, some more specialized provers, especially one dedicated to projective geometry [5, 6, 11], have been recently developed in the IGG team.

2.2 Research project and challenges

Formal proof developments (proof scripts) developped in Coq or other modern proof assistants such as Lean [9] or Isabelle/HOL [13] get bigger and bigger[1]. They often embed various specific tools to handle some pieces of the proofs automatically. Hence writing the initial proof development, maintaining it and compiling (= checking the correctness of) it becomes more and more challenging. In this thesis project, we propose to work in two directions to help writing the proofs and maintaining them once they are written.

First, we shall enhance the communication between the automated provers and Coq (by transforming the proof traces they produce into proof certificates automatically checkable by Coq). This could be achieved throught the approach proposed in [1, 10]. A first application would consist in making the prover based on projective geometry inter-operate with Coq. The automated tool could then be directly called from a proof session inside the Coq proof assistant and shall produce a Coq proof script which enables to go one step further in the proof. The next step would then be to integrate this reasoning mechanisms directly inside the Coq proof assistant and provide them as interactive tactics, either implemented as Ocaml plugins or using using some meta-programming tools such as Coq-elpi [19] or MetaCoq [18].

Second, as proofs grow in size, technicality and complexity, reusing an older formal Coq proof script is often quite difficult especially because successive versions of Coq may lead to some incompatibilities. This is mainly caused by the fact that Coq is a research tool and remains under heavy development. In this project, we propose to design and implement new tools to help making completed proof developments more resilient and more easily reusable in newer versions of Coq. We aim at proposing some automatic proof scripts transformations to identify and then fix the main weaknesses of Coq proof scripts. This ranges from detecting the use of a non-properly introduced variable to enhancing the structure of the proof scripts. Another application could be to automatically replace some tactics with some equivalent ones (w.r.t the considered goals) which are faster, simpler, non deprecated yet, etc. All these examples of *proof engineering* could be used in conjunction with the recent works of Talia Ringer [17, 16, 15] on *proof repair*. Eventually, *proof engineering* should reach the same standards as *sofware engineering*, making proofs more reliable, more easily reusable and thus easier to deal with by non-specialists.
In the longer run, the proposed automatic transformation tools for formal proofs could be extended to automatically transform some proof scripts from one (older) version of Coq to a newer one. It could also be used to carry out proof transformations from one proof assistant, e.g. Lean or Isabelle/HOL to another as it is done in Dedukti [2]. This work would integrate smoothly in the European research network on digital proofs[2] led by Frédéric Blanqui (Laboratoire de Méthodes Formelles[3], ENS Paris-Saclay).

To evaluate the performances of these tools, we shall apply them to large proof developements such as those underway in our research team. We shall especially focus on the study of finite models of projective geometry where the size of the models makes the formalization in an interactive system like Coq very challenging, pursing some recent work on the smallest projective space PG(3,2) [12]. Other applications are envisioned in the field of combinatorics of geometric objects. Among them, we shall consider enumerating some families of combinatorial maps, and establish some non-trivial equivalences between some families of maps and some families of lambda-terms like those presented in [20]. We are also interested in applying our expertise to other areas of mathematics.

[1]Check out some examples at `https://github.com/coq-community` or `https://www.isa-afp.org/`
[2]`https://europroofnet.github.io/`
[3]`https://lmf.cnrs.fr/`

References

[1] Michaël Armand, Germain Faure, Benjamin Grégoire, Chantal Keller, Laurent Théry, and Benjamin Werner. A modular integration of SAT/SMT solvers to coq through proof witnesses. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs - First International Conference, CPP 2011, Kenting, Taiwan, December 7-9, 2011. Proceedings*, volume 7086 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2011.

[2] Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Gilles Dowek, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard. Dedukti: a logical framework based on the $\lambda - \Pi$-calculus modulo theory. Manuscript `http://www.lsv.fr/~dowek/Publi/expressing.pdf`, 2016.

[3] Clark W. Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 171–177. Springer, 2011.

[4] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions.* Springer, 2004.

[5] David Braun. *Approche combinatoire pour l'automatisation en Coq des preuves formelles en géométrie d'incidence projective.* PhD thesis, Université de Strasbourg, sept. 2019.

[6] David Braun, Nicolas Magaud, and Pascal Schreck. Two new ways to formally prove dandelin-gallucci's theorem. In Frédéric Chyzak and George Labahn, editors, *ISSAC '21: International Symposium on Symbolic and Algebraic Computation, Virtual Event, Russia, July 18-23, 2021*, pages 59–66. ACM, 2021.

[7] Coq development team. *The Coq Proof Assistant Reference Manual, Version 8.14.0*, 2021.

[8] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[9] Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean Theorem Prover (System Description). In *Proceedings of CADE 2015*, volume 9195 of *LNCS*, pages 378–388. Springer, 2015.

[10] Burak Ekici, Alain Mebsout, Cesare Tinelli, Chantal Keller, Guy Katz, Andrew Reynolds, and Clark W. Barrett. Smtcoq: A plug-in for integrating SMT solvers into coq. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 126–133. Springer, 2017.

[11] Nicolas Magaud. Integrating an automated prover for projective geometry as a new tactic in the coq proof assistant. In Chantal Keller and Mathias Fleury, editors, Proceedings Seventh Workshop on *Proof eXchange for Theorem Proving,* Pittsburg, USA, 11th July 2021, volume 336 of *Electronic Proceedings in Theoretical Computer Science*, pages 40–47. Open Publishing Association, 2021.

[12] Nicolas Magaud. Proof pearl: Formalizing spreads and packings of the smallest projective space pg(3, 2) using the coq proof assistant. In June Andronick and Leonardo de Moura, editors, *13th International Conference on Interactive Theorem Proving, ITP 2022, August 7-10, 2022, Haifa, Israel*, volume 237 of *LIPIcs*, pages 25:1–25:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[13] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[14] Alexandre Riazanov and Andrei Voronkov. The design and implementation of VAMPIRE. *AI Commun.*, 15(2-3):91–110, 2002.

[15] Talia Ringer. *Proof Repair.* PhD thesis, University of Washington, 2021.

[16] Talia Ringer, Karl Palmskog, Ilya Sergey, Milos Gligoric, and Zachary Tatlock. QED at large: A survey of engineering of formally verified software. *Found. Trends Program. Lang.*, 5(2-3):102–281, 2019.

[17] Talia Ringer, Nathaniel Yazdani, John Leo, and Dan Grossman. Adapting proof automation to adapt proofs. In June Andronick and Amy P. Felty, editors, *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8-9, 2018*, pages 115–129. ACM, 2018.

[18] Matthieu Sozeau, Abhishek Anand, Simon Boulier, Cyril Cohen, Yannick Forster, Fabian Kunze, Gregory Malecha, Nicolas Tabareau, and Théo Winterhalter. The metacoq project. *J. Autom. Reason.*, 64(5):947–999, 2020.

[19] Enrico Tassi. Elpi: an extension language for Coq (Metaprogramming Coq in the Elpi λProlog dialect). The Fourth International Workshop on Coq for Programming Languages: CoqPL 2018, January 2018.

[20] Noam Zeilberger and Alain Giorgetti. A correspondence between rooted planar maps and normal planar lambda terms. *Logical Methods in Computer Science*, 11(3), 2015.