Sujet de stage de M2 Informatique : Robustesse et Maintenance des preuves formelles écrites en Coq

Please check out page 2 for an english version of this document.

Nicolas Magaud (magaud@unistra.fr)
https://dpt-info.u-strasbg.fr/~magaud/
Lab. ICube UMR 7357 CNRS Université de Strasbourg, France

L'outil d'aide à la preuve Coq [3, 2] est de plus en plus utilisé et les développements deviennent de plus en plus imposants en taille ainsi qu'en temps de compilation [1]. Réutiliser un développement effectué il y a plusieurs années est souvent difficile à cause de problèmes de compatibilité inhérents au fait que Coq reste un outil de recherche dont le développement est toujours très actif.

Dans ce cadre de ce stage, nous souhaitons proposer de nouveaux outils permettant d'assurer la pérennité des développements formels en les rendant plus robustes une fois terminée par l'utilisateur. Il s'agira d'effectuer des transformations automatiques sur les scripts de preuve produits pour détecter les points fragiles de ces scripts. On pourra, par exemple, détecter et éventuellement supprimer l'utilisation par une tactique d'une variable non explicitement introduite précédemment. On pourra aussi renforcer la structure du script de preuve en imposant l'utilisation des *bullets* ou bien des accolades pour mettre en avant la structure de la preuve. Une autre application possible sera de substituer des tactiques par d'autres (équivalentes sur le but considéré), mais plus efficaces que ce soit en temps d'exécution/compilation ou en espace mémoire utilisé). D'autres exemples de transformations pourront être inspirés des travaux récents de Talia Ringer [6, 5, 4] sur la notion de réparation des preuves *Proof repair*.

A plus long terme, les outils de transformation automatique de preuves proposées pourraient être utilisés pour transformer automatiquement des preuves d'une version de Coq à une autre, ou plus généralement vers un autre assistant de preuve comme cela se fait dans Dedukti [1].

Références

[1] 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.

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

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

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

[5] 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.

[6] 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.

1. Voir quelques exemples ici : `https://github.com/coq-community`

Master Internship in Computer Science
Making Coq Proofs More Reliable and More Easily Reusable

Nicolas Magaud (magaud@unistra.fr)
https://dpt-info.u-strasbg.fr/~magaud/
Lab. ICube UMR 7357 CNRS Université de Strasbourg, France

Proof assistants such as Coq [3, 2] are increasingly used to carry out formal proof developments. These proofs get bigger and bigger [1]. Besides, compiling such proof developments may also be challenging.

Reusing a 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 internship, 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 eventually fix the main weaknesses of Coq proof scripts. This ranges from detecting the use of a variable with having explicitly introduced it beforehand to enhancing the structure of the proof scripts by using the *bullets* or curly brackets features of the proof language. 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. Other examples of applications can be retrieved from the recent works of Talia Ringer [6, 5, 4] on *proof repair*.

In the longer run, these 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 to another as it is done in Dedukti [1].

Références

[1] 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.

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

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

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

[5] 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.

[6] 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.

1. Check out some examples here : https://github.com/coq-community