Les Anciens et les Modernes: Communication entre Coq et Lean

Proposition de stage de L3

Chantal KELLER <Chantal.Keller@lri.fr>

1 Cadre du stage

Le stage se déroulera au Laboratoire de Recherche en Informatique de l'Université Paris-Sud, sous la direction de Chantal KELLER.

Il peut être gratifié.

2 Contexte

Coq [1] est un assistant de preuve utilisé dans nombre d'applications, allant de la formalisation de propriétés mathématiques [GAA+13] à la certification d'un compilateur optimisant pour C [Ler09]. Il s'agit d'un assistant de preuve mature, dont la première implantation remonte aux années 1980 : il dispose ainsi de très nombreuses bibliothèques et applications [2], de multiples extensions, d'une interface très riche, ... Cela lui offre une très grande visibilité et reconnaissance dans le domaine des preuves formelles et des langages de programmation, concrétisée en 2013 par le prix "ACM SIGPLAN Programming Languages Software".

Malheureusement, cette maturité s'accompagne d'un héritage présentant un certain nombre d'inconvénients : par exemple, le code du noyau de Coq est devenu assez large, ce qui diminue la confiance dans cet assistant de preuves. Par ailleurs, le fait d'avoir un très grand nombre d'utilisateurs ralentit l'ajout d'améliorations dans le système, pour éviter à ces utilisateurs le coût d'adaptation de leurs développements.

Lean [3] est un assistant de preuve basé sur les mêmes fondements théoriques que Coq, dont la première implantation remonte à 2013. En constante évolution, il intègre facilement et rapidement de nouvelles fonctionnalités : ainsi, il offre par exemple une bonne automatisation, différents modes pour effectuer des preuves, il supporte différentes logiques, ... Pour ces raisons, il connaît actuellement un très grand intérêt dans la communauté des preuves formelles.

Cependant, cette jeunesse fait que Lean dispose d'une bibliothèque et de développements beaucoup moins riches que Coq. Par ailleurs, l'évolution rapide rend encore risqué le fait de développer de grands projets en Lean, car ceux-ci nécessiteraient une adaptation constante aux modifications du système.

Coq et Lean pourraient bénéficier à la fois des avantages de la jeunesse et de ceux de la maturité, par la communication. Ainsi, l'importation automatique dans Lean des théorèmes prouvés en Coq permettrait au premier de bénéficier de toutes les bibliothèques et applications

1. https://coq.inria.fr
2. https://coq.inria.fr/opam/www
3. http://leanprover.github.io

qui ont été formalisées dans le dernier. À l'inverse, l'importation automatique dans Coq des théorèmes prouvés en Lean permettrait au premier de bénéficier des avantages du dernier, comme son automatisation, sans craindre de devoir suivre l'évolution du système.

En conclusion, cette double communication permettrait de bénéficier des avantages des deux systèmes lors de formalisations. Parlant le même langage, elle ne s'accompagnerait pas d'un surcoût dû à une traduction exigeante, comme cela est le cas pour des systèmes plus éloignés [KW10, AFG+11].

3 Contribution attendue

Le but du stage est d'étudier et de mettre en place une première passerelle de communication entre ces deux assistants de preuve pour l'échange de théorèmes, qui permettra aux utilisateurs de mêler Coq et Lean dans leurs formalisations. Cette passerelle, écrite en OCaml, devra traduire automatiquement les énoncés des théorèmes entre les deux systèmes.

Plus qu'une implantation en OCaml, ce stage nécessite d'analyser les représentations logiques des deux systèmes, et d'en comprendre les similitudes et les différences, afin de produire une traduction correcte tout en étant la plus complète possible.

4 Perspectives

Ce travail ouvrira la porte vers une traduction automatique *sceptique* des théorèmes, dans laquelle non seulement leurs énoncés sont échangés, mais également leurs preuves, afin d'obtenir une formalisation vérifiable complètement et indifféremment dans l'un ou l'autre système. Ces aspects pourront être abordés par le candidat selon ses goûts.

5 Prérequis

Des connaissances basiques en théorie de la démonstration sont attendues (correspondance de Curry-Howard). Il n'est pas nécessaire de savoir développer des preuves en Coq ou en Lean.

Références

[AFG+11] 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, *CPP*, volume 7086 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 2011.

[GAA+13] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-Checked Proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2013.

[KW10] Chantal Keller and Benjamin Werner. Importing HOL light into coq. In Matt Kaufmann and Lawrence C. Paulson, editors, *Interactive Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6172 of *Lecture Notes in Computer Science*, pages 307–322. Springer, 2010.

[Ler09] Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7) :107–115, 2009.