Calcul vérifiable pour Coq

Proposition de stage de Master

Chantal KELLER <Chantal.Keller@lri.fr>

October, 24th 2016

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.

2 Contexte

Coq est un assistant de preuve basé sur la Théorie des Types. Le calcul est une notion centrale de ce formalisme logique : la règle de conversion permet d'identifier les preuves de deux énoncés équivalents modulo les règles de calcul :

$$\frac{\Gamma \vdash p : A \qquad A \equiv B}{\Gamma \vdash p : B}\ \textsc{Conv}$$

Cette possibilité d'utiliser le calcul au sein de la logique est notamment mise en avant par la *réflexion calculatoire* [ACHA90], qui consiste à remplacer une partie d'un arbre de preuve par du calcul. Ce procédé est utilisé en Coq par exemple pour vérifier la primalité d'entiers [GTW06], prouver des équations dans des anneaux [GM05], certifier des prouveurs automatiques [AFG+11], ou encore dans la preuve du Théorème des quatre couleurs [Gon07].

Cela a conduit à implanter dans le noyau de Coq divers algorithmes performants pour effectuer du calcul, notamment une machine virtuelle [GL02] et un mécanisme de compilation vers du code natif via le langage OCaml [BDG11]. L'objectif du stage est d'étudier comment intégrer à Coq un nouveau mécanisme de calcul.

Ce mécanisme est basé sur le *calcul vérifiable*, un procédé cryptographique permettant de déléguer un calcul à une machine potentiellement non fiable, en lui demandant de fournir un certificat permettant de vérifier en temps constant que les calculs ont été effectués correctement avec une garantie très forte reposant sur des arguments cryptographiques. Un exemple de protocole pour le calcul vérifiable est Pinocchio [PHGR13], dont Gepetto [CFH+15] fournit un compilateur depuis le langage C et un vérificateur de certificats.

L'intégration de ce mécanisme à Coq offrirait les possibilités suivantes :

— améliorer l'efficacité du calcul ;

— permettre de réutiliser des calculs par un mécanisme de cache.

Par exemple, une preuve du Théorème des quatre couleurs accompagnée du certificat pourrait être vérifiée en quelques secondes avec une très grande confiance.

3 Contribution attendue

Le stage comporte des aspects théoriques et d'implantation. La contribution attendue est la suivante :

— définir un algorithme de compilation d'un langage fonctionnel pur vers le format d'entrée de Pinocchio ;
— implanter cet algorithme en OCaml et le lier à Coq et à Gepetto ;
— tester l'efficacité sur un choix de benchmarks.

Selon l'évolution du stage et les compétences du candidat, les extensions suivantes sont possibles :

— implanter le cache des certificats ;
— vérifier formellement en Coq le compilateur (en se basant sur une certification du compilateur de Gepetto [FKL16]) ;
— étendre le compilateur aux types dépendants.

4 Prérequis

Des connaissances en théorie des langages de programmation sont attendues (sémantique, compilation). Des connaissances basiques en cryptographie sont les bienvenues. Il n'est pas nécessaire de connaître l'assistant de preuve Coq.

Références

[ACHA90] Stuart F. Allen, Robert L. Constable, Douglas J. Howe, and William E. Aitken. The Semantics of Reflected Proof. In *LICS*, pages 95–105. IEEE Computer Society, 1990.

[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 Jouannaud and Shao [JS11], pages 135–150.

[BDG11] Mathieu Boespflug, Maxime Dénès, and Benjamin Grégoire. Full Reduction at Full Throttle. In Jouannaud and Shao [JS11], pages 362–377.

[CFH+15] Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto : Versatile Verifiable Computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 253–270. IEEE Computer Society, 2015.

[FKL16] Cédric Fournet, Chantal Keller, and Vincent Laporte. A Certified Compiler for Verifiable Computing. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 268–280. IEEE Computer Society, 2016.

[GL02] Benjamin Grégoire and Xavier Leroy. A compiled implementation of strong reduction. In Mitchell Wand and Simon L. Peyton Jones, editors, *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming (ICFP '02), Pittsburgh, Pennsylvania, USA, October 4-6, 2002.*, pages 235–246. ACM, 2002.

[GM05] Benjamin Grégoire and Assia Mahboubi. Proving Equalities in a Commutative Ring Done Right in Coq. In Joe Hurd and Thomas F. Melham, editors, *TPHOLs*, volume 3603 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2005.

[Gon07] Georges Gonthier. The Four Colour Theorem : Engineering of a Formal Proof. In Deepak Kapur, editor, *Computer Mathematics, 8th Asian Symposium, ASCM 2007,*

Singapore, December 15-17, 2007. Revised and Invited Papers, volume 5081 of *Lecture Notes in Computer Science*, page 333. Springer, 2007.

[GTW06] B. Grégoire, L. Théry, and B. Werner. A computational approach to Pocklington certificates in type theory. In M. Hagiya and Ph. Wadler, editors, *FLOPS*, volume 3945 of *Lecture Notes in Computer Science*, pages 97–113. Springer, 2006.

[JS11] 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*. Springer, 2011.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio : Nearly Practical Verifiable Computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252. IEEE Computer Society, 2013.