Systèmes linéaires, arithmétique par intervalles et calcul certifié en Coq

Proposition de stage de L3

Équipe ACADIE, Lab. IRIT, Université Paul Sabatier
Équipe DTIS/CoVNI, ONERA, Toulouse

Janvier 2022

Informations pratiques Le stage se déroulera à l'*Institut de Recherche en Informatique de Toulouse* (IRIT) sur le campus de l'*Université Paul Sabatier* (UPS).
Il sera encadré par Érik MARTIN-DOREL (erik.martin-dorel@irit.fr) et Pierre ROUX (pierre.roux@onera.fr) (ONERA).

Contexte Les assistants de preuve sont des logiciels complexes qui permettent d'encoder des théories mathématiques et des algorithmes dans un langage formel, de formuler des propriétés et des théorèmes sur ces algorithmes, de développer des preuves pour ces propriétés et de vérifier mécaniquement ces preuves formalisées avec un très haut niveau de garantie.

Grâce à sa logique sous-jacente très expressive, l'assistant de preuve Coq offre un support intégré pour les calculs. Il propose plusieurs tactiques de réduction comme `compute`, `vm_compute` [GL02] et `native_compute` [BDG11] qui peuvent tirer parti de la compilation vers du bytecode ou du code natif OCaml, de façon à améliorer la performance des calculs considérés, au prix d'une base de confiance légèrement plus grande.

Couplé avec les fonctionnalités de preuve par réflexion de Coq, ces tactiques de réduction ont joué un rôle clef dans de nombreux travaux de certification formelle de calculs numériques ou symboliques.

Un exemple de ces travaux est donné par la bibliothèque CoqInterval [Mel12, MDM15] : une tactique réflexive pour prouver formellement et automatiquement des inégalités sur des expressions à variable réelle. Cette tactique s'appuie sur des algorithmes d'arithmétique d'intervalles, liés à un usage systématique des arrondis dirigés (vers $-\infty$ et $+\infty$), définis dans la norme IEEE754 [IEE08].

Des travaux récents dans CoqInterval ont permis d'accroître les performances des opérations flottantes, en utilisant directement en Coq les opérations en virgule flottante du processeur [BMR19] (en arrondi au plus plus proche), plutôt que d'émuler ces opérations flottantes en utilisant des nombres entiers.

Nous visons désormais à étudier l'application de ces travaux à la formalisation de méthodes de calcul numérique rigoureux en algèbre linéaire [Rum10], en commençant par mécaniser en Coq certaines de ces méthodes.

Contribution attendue Ce stage consiste à étudier formellement l'article de "survey" consacré par Rump au calcul flottant rigoureux [Rum10], en se focalisant essentiellement sur la Section 10, liée aux systèmes linéaires. Après s'être familiarisé avec le domaine d'étude, on s'attachera à implémenter (en OCaml/Coq) l'Algorithme 10.7, puis à étudier son fonctionnement sur plusieurs cas d'étude. L'implémentation Coq pourra donner lieu à une spécification formelle de cet algorithme, en s'aidant des bibliothèques Flocq [BM11], CoqInterval, et MathComp [MT21], ainsi que des preuves papier données dans l'article.

Pré-requis Le candidat doit avoir un intérêt pour la programmation fonctionnelle et la logique mathématique. Une expérience préalable de Coq ou d'un autre assistant de preuve serait appréciée mais n'est pas nécessaire. Aucune connaissance des librairies Coq citées dans ce sujet (Flocq, CoqInterval, MathComp) ni de l'arithmétique flottante ou de l'arithmétique d'intervalle n'est attendue.

Références

[BDG11] Mathieu Boespflug, Maxime Dénès, and Benjamin Grégoire. Full reduction at full throttle. In Jean-Pierre Jouannaud and Zhong Shao, editors, *CPP 2011. Proceedings*, volume 7086 of *LNCS*, pages 362–377. Springer, 2011. doi:10.1007/978-3-642-25379-9_26.

[BM11] Sylvie Boldo and Guillaume Melquiond. Flocq : A Unified Library for Proving Floating-Point Algorithms in Coq. In Elisardo Antelo, David Hough, and Paolo Ienne, editors, *ARITH 2011*, pages 243–252. IEEE, 2011. doi:10.1109/ARITH.2011.40.

[BMR19] Guillaume Bertholon, Érik Martin-Dorel, and Pierre Roux. Primitive Floats in Coq. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *ITP 2019*, volume 141 of *LIPIcs*, pages 7 :1–7 :20. Schloss Dagstuhl – LZI, 2019. doi:10.4230/LIPIcs.ITP.2019.7.

[GL02] Benjamin Grégoire and Xavier Leroy. A compiled implementation of strong reduction. In Mitchell Wand and Simon L. Peyton Jones, editors, *ICFP 2002*, pages 235–246. ACM SIGPLAN, 2002. doi:10.1145/581478.581501.

[IEE08] IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754–2008*, August 2008. doi:10.1109/IEEESTD.2008.4610935.

[MDM15] Érik Martin-Dorel and Guillaume Melquiond. Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq. *Journal of Automated Reasoning*, pages 1–31, 2015. doi:10.1007/s10817-015-9350-4.

[Mel12] Guillaume Melquiond. Floating-point arithmetic in the Coq system. *Inf. Comput.*, 216 :14–23, 2012. doi:10.1016/j.ic.2011.09.005.

[MT21] Assia Mahboubi and Enrico Tassi. *Mathematical Components*. Zenodo, Jan 2021. doi:10.5281/zenodo.4457887.

[Rum10] Siegfried Rump. Verification methods : Rigorous results using floating-point arithmetic. *Acta Numerica*, 19 :287–449, May 2010. doi:10.1017/S096249291000005X.