Compilation optimisante et certifiée pour la précision numérique

Lieu : Université de Perpignan Via Domitia.

Equipe d'accueil : Equipe Projet DALI, laboratoire LIRMM (UMR 5506).

Encadrants : David Delmas (Airbus) et Matthieu Martel (LIRMM).

Durée : 4 à 6 mois.

Contacts : `david.delmas@airbus.com` et `matthieu.martel@univ-perp.fr`.

Mots clés : Interprétation abstraite, Transformation sémantique de Programmes, Compilation, Précision Numérique, Nombres flottants, IEEE754.

Les erreurs d'arrondi sont indissociables des calculs sur ordinateurs dans lesquels le résultat de chaque opération doit être approché pour être représenté en mémoire sur un nombre fini de bits. Ces erreurs sont en général acceptables dans la mesure où le résultat produit par la machine est proche de celui que l'on aurait obtenu en utilisant des nombres réels. Cependant, elles peuvent aussi être amplifiées par propagation, dénuant de sens le résultat d'un calcul. Aussi, est-il souhaitable d'optimiser les traitements numériques présents dans des programmes au moment de la compilation, afin que le code généré produise des résultats aussi proche que possible de ceux que l'on obtiendrait si les calculs étaient effectués avec des nombres réels.

Actuellement, les outils tels que Fluctuat [4], permettant d'estimer ou de borner les erreurs d'arrondi survenant au cours d'un calcul réalisé sur ordinateur, rencontrent un vif succès. Cependant, ces techniques n'indiquent pas comment réduire les erreurs ainsi mesurées. Or, améliorer manuellement la précision numérique d'un programme est tout à la fois fastidieux et difficile, car l'arithmétique utilisée est particulièrement non-intuitive (constantes non représentables exactement, opérations non-associatives, non-inversibles, etc.)

Ce stage concerne une nouvelle sorte de transformation automatique de programmes permettant d'optimiser la précision des calculs au moment de la compilation : si l'on suppose qu'un programme fonctionnerait correctement s'il était exécuté en utilisant des nombres réels, il est possible de le modifier automatiquement, par des méthodes de transformation de codes, afin de produire un autre programme sémantiquement équivalent et numériquement plus précis [5] (autrement dit, d'améliorer sa précision). Pour cela, on autorise des ré-écritures de formules, normalement fausses dans l'arithmétique de la machine (associativité, factorisation, modification de constantes) de façon à générer automatiquement une nouvelle formule, mathématiquement égale à la première et plus précise en ce qui concerne son évaluation par un ordinateur. Afin de maitriser l'explosion combinatoire du nombre de ré-écritures, celles-ci sont décrites à l'aide de sémantiques abstraites conformément à la théorie de l'interprétation abstraite [2, 3]. De telles transformations ont été implémentées dans le logiciel Sardana qui permet actuellement d'optimiser la précision des calculs présents dans des codes Lustre/SCADE interfacés avec du C.

L'objectif de ce stage concerne la certification des transformations produites par Sardana. Afin de pouvoir être intégrées à des logiciels critiques, les transformations proposées doivent être garanties. Une première approche consisterait à certifier l'outil lui-même, ce qui serait fastidieux. Au lieu de cela nous proposons de certifier directement le résultat qu'il propose, indépendamment de la façon dont il a été produit. Pour cela il est nécessaire de générer une preuve formelle de l'équivalence mathématique (dans les réels) entre le programme initial et le programme transformé (en reconstituant le cheminement de Sardana ayant produit la transformation). Ce certificat (qui pourra être par exemple un script Gappa [1]) devra pouvoir être vérifié par un démonstrateur de théorèmes et sera complétée par le calcul, par analyse statique, de la borne d'erreur entre le calcul flottant et le calcul exact. La combinaison de ces deux éléments permettra d'affirmer que le nouveau programme est mathématiquement équivalent au premier tout en étant plus précis.

Ce stage a pour vocation d'être poursuivi dans le cadre d'une thèse CIFRE qui se déroulera pour moitié au sein de l'équipe DALI (LIRMM-UPVD) et pour moitié dans les locaux d'Airbus, afin de permettre au doctorant de travailler au plus près des processus de développement et de définir une méthodologie en rapport avec le contexte industriel.

Références

[1] Sylvie Boldo, Jean-Christophe Filliâtre, and Guillaume Melquiond. Combining coq and gappa for certifying floating-point programs. In *Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009*, volume 5625 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2009.

[2] P. Cousot and R. Cousot. Abstract interpretation : A unified lattice model for static analysis of programs by construction of approximations of fixed points. In *Principles of Programming Languages 4*, pages 238–252. ACM Press, 1977.

[3] P. Cousot and R. Cousot. Systematic design of program transformation frameworks by abstract interpretation. In *Conference Record of the Twentyninth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 178–190, Portland, Oregon, 2002. ACM Press, New York, NY.

[4] D. Delmas, E. Goubault, S. Putot, J. Souyris, K. Tekkal, and F. Vedrine. Towards an industrial use of fluctuat on safety-critical avionics software. In *Formal Methods for Industrial Critical Systems, 14th International Workshop*, FMICS, 2009.

[5] M. Martel. Enhancing the implementation of mathematical formulas for fixed-point and floating-point arithmetics. *Journal of Formal Methods in System Design*, 35 :265–278, 2009.