PREUVE DE PROGRAMMES : GÉNÉRATION D'INVARIANTS POLYNOMIAUX

ÉQUIPE CELTIQUE
IRISA/INRIA RENNES BRETAGNE ATLANTIQUE
DAVID CACHERA ARNAUD JOBIN FLORENT KIRCHNER

MOTS-CLÉS : Preuve de programmes, invariants, analyse statique, bases de Gröbner et algorithmes associés.

Les méthodes d'analyse statique, connues pour leurs applications dans le domaine de la sécurité et de la sûreté de fonctionnement, sont depuis une décennie testées sur des propriétés de programmes *quantitatives*, spatiales et temporelles [4, 2]. Ces nouvelles applications ne vont pas sans poser, néanmoins, de sérieuses difficultés.

Tentons d'illustrer ce propos, avec le problème de la génération d'invariants polynomiaux. Traditionnellement, les méthodes d'analyse de programmes requièrent d'annoter certaines parties de code (boucles, branchements, appels de fonction, etc.) par les propriétés logiques qu'ils observent ; on appelle ces annotations des *invariants*. Spécifier des invariants à la main est une tâche complexe sinon laborieuse, et son automatisation peut être vue comme l'une des clés de voûte de la dissémination des méthodes formelles en ingénierie logicielle. Si le cas d'invariants linéaires a été résolu assez tôt [3], il reste difficile dans le cas non-linéaire. En somme, la question peut se résumer comme le problème ouvert suivant :

> « Comment trouver des invariants polynomiaux de programmes, automatiquement et aussi précisément que possible ? »

Ce n'est que très récemment que quelques premières solutions ont été proposées [6, 7, 8]. Ces approches reposent sur l'utilisation d'idéaux de polynômes pour calculer une version abstraite du comportement des programmes. Cependant les approches existantes souffrent de certaines limitations qui empêchent la découverte d'invariants pertinents ; l'équipe Celtique se propose d'utiliser un ensemble de techniques avancées pour améliorer ce type d'analyse. Les briques de base de ce travail — opérations sur les idéaux, itérations de plus grand point fixe — étant maintenant bien définies, il reste à confronter les définitions théoriques avec une implémentation pratique. L'expérimentation permettra, en retour, d'optimiser les algorithmes d'analyse, en particulier du point de vue de leur efficacité.

Le but de ce stage est de participer à l'implémentation en Caml du prototype de recherche en question. Une première piste de travail pourrait s'orienter vers l'adaptation d'algorithmes de calcul de bases de Gröbner [1, 5] aux contraintes émergeant de cette nouvelle analyse, notamment en tenant compte de son caractère incrémental.

Ce sujet requiert à la fois de manipuler des concepts mathématiques avancés, et de relever le défi d'efficacité attaché à l'implémentation des algorithmes associés. Le travail sera conduit en liaison serrée avec une équipe de recherche jeune et dynamique : le candidat devra démontrer une forte capacité d'innovation, et un goût prononcé pour les interactions scientifiques.

Références

[1] Bruno Buchberger. Gröbner bases : A short introduction for systems theorists. *Lecture Notes in Computer Science*, 2178 :1–10, 2001.

[2] David Cachera, Thomas P. Jensen, Arnaud Jobin, and Pascal Sotin. Long-run cost analysis by approximation of linear operators over dioids. *Mathematical Structures in Computer Science*, 20(4) :589–624, 2010.

[3] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 84–96, Tucson, Arizona, January 1978.

[4] Alessandra Di Pierro and Herbert Wiklicky. Concurrent constraint programming : towards probabilistic abstract interpretation. In *PPDP*, pages 127–138, 2000.

[5] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In Teo Mora, editor, *ISSAC'2002*, pages 75–83. ACM Press, 2002.

[6] Markus Müller-Olm and Helmut Seidl. Computing polynomial program invariants. *Inf. Process. Lett*, 91(5) :233–244, 2004.

[7] Enric Rodríguez-Carbonell and Deepak Kapur. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Sci. Comput. Program.*, 64 :54–75, January 2007.

[8] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Non-linear loop invariant generation using Gröbner bases. *ACM SIGPLAN Notices*, 39(1) :318–329, January 2004.