Proposition de stage Master 2 recherche

Génération de graphes de dépendances de programmes modulaires

Mots-clés : méthodes formelles, analyse statique, vérification de programmes, programmes C

Cadre

Le CEA LIST est un centre de recherche technologique sur les systèmes à logiciel prépondérant qui mène ses recherches en partenariat avec les grands acteurs industriels du nucléaire, de l'automobile, de l'aéronautique, de la défense et du médical pour étudier et développer des solutions innovantes adaptées à leurs besoins. Au sein du CEA LIST, le Laboratoire de Sûreté des Logiciels (LSL), localisé à Saclay (Essonne, 91), développe des outils d'aide à la validation et à la vérification de logiciels et de systèmes matériels/logiciels, tout particulièrement dans le domaine des systèmes embarqués critiques.

L'un des nos outils, nommé Frama-C (`http://frama-c.com`), est une plate-forme logicielle facilitant le développement d'outils d'analyses statiques de programmes C. Le stage se déroulera au sein de l'équipe de R&D développant Frama-C, qui est programmé en *OCaml*.

Objectifs

Frama-C propose un certain nombre d'analyses visant à faciliter la compréhension d'un code C. On citera le *slicing*, qui permet notamment de supprimer d'un programme toutes les instructions sans importance pour le calcul d'une variable choisie par l'utilisateur, ou *l'analyse d'impact* qui détermine toutes les instructions dont l'évaluation dépend des effets d'une instruction donnée. Ces deux fonctionnalités utilisent en interne un composant appelé *PDG* (pour *Program Dependence Graph* [FOW87]), qui est calculé automatiquement par Frama-C à partir des résultats de l'interpréteur abstrait de la plate-forme, Value [CKK+12, §3]. Ces graphes sont corrects par construction, car Value calcule une sur-approximation de toutes les valeurs possibles à l'exécution, y compris pour les variables pointeurs. Dès lors, toutes les interférences possibles entre deux pointeurs potentiellement aliasés sont prises en compte.

Cette approche garantit l'obtention de PDGs corrects, mais présente toutefois un inconvénient : les PDGs obtenus sont non modulaires. Un pointeur n'est pas représenté symboliquement, mais plutôt par l'ensemble des locations sur lesquelles il peut pointer. Dès lors, dans le programme `*p=v; z=*p;`, si lors de l'analyse il est déterminé que `p` peut pointer sur `x` ou `y`, le PDG va indiquer que la valeur de `z` est `v`, mais potentiellement également les valeurs initiales de `x` ou `y`. En un sens, les informations d'un tel PDG sont non relationnelles. On pourra également se convaincre que sous l'environnement initial `p == &p`, le PDG obtenu aurait été très différent. La construction de PDGs modulaires est non triviale dans des langages avec pointeurs, plus encore si le typage peut être contourné, comme en C : générer des PDGs corrects dans tous les cas conduit en général à un résultat imprécis.

La solution intermédiaire que nous proposons consiste à utiliser une analyse d'alias, qui fournirait une sur-approximation des contextes d'appels possibles

pour les fonctions dépendant de pointeurs, notamment au niveau des conditions d'aliasing. Dès lors, il devient possible de générer un PDG modulaire qui correspondrait à ces hypothèses d'aliasing «réelles», ou même plusieurs PDGs, correspondant chacun à des contextes distincts. Ces PDGs seraient alors munis d'une précondition logique exprimant l'aliasing possible en entrée de la fonction pour que le PDG soit correcte.

La stage consistera d'une part à définir et générer des PDGs semi-modulaires (dans lesquels les pointeurs sont représentés symboliquement), qui généralisent les PDGs actuels de Frama-C. Ces PDGs seront générés sous certaines hypothèses d'aliasing, supposées données en entrée. En parallèle, on s'intéressera à l'identification automatique de ces hypothèses d'alias, soit à partir des résultats fournis par Value, soit à partir d'une analyse plus simple telle que celle de Steensgard [Ste96].

Candidatures

Profil des candidats

- Bonnes connaissances en OCaml
- Intérêt pour l'analyse automatique de programmes réels
- Capacité de travail en équipe
- Une connaissance du C serait un plus.

Conditions : stage indemnisé, aide au logement possible, transports CEA en Île-de-France.

Contact : Boris Yakobowski (*prénom.nom@cea.fr*)

Les délais administratifs au CEA étant de 2 à 3 mois minimum, merci de prendre contact le plus tôt possible.

Références

[CKK+12] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C - a software analysis perspective. In George Eleftherakis, Mike Hinchey, and Mike Holcombe, editors, *SEFM*, volume 7504 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2012.

[FOW87] Jeanne Ferrante, Karl J. Ottenstein, and Joe D. Warren. The program dependence graph and its use in optimization. *ACM Trans. Program. Lang. Syst.*, 9(3) :319–349, July 1987.

[Ste96] Bjarne Steensgaard. Points-to analysis in almost linear time. In *Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '96, pages 32–41, New York, NY, USA, 1996. ACM.