Proposition de stage niveau bac+5

Vérification Dynamique de Propriétés Mémoires Subtiles sur des programmes C

Mots-clés : *runtime assertion checking*, compilation

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 de programmes C. Un de ces outils, nommé *E-ACSL*, est dédié à la vérification dynamique de propriétés (*runtime verification*). Le stage se déroulera au sein de l'équipe de R&D développant *Frama-C* et, plus spécifiquement, *E-ACSL*. Ces outils sont développés en *OCaml*.

Objectifs

Chaque programme C analysé par *Frama-C* peut être annoté par des spécifications formelles, écrites dans un langage appelé *ACSL*. *Frama-C* offre alors différentes techniques de vérification pour garantir que le programme satisfait sa spécification. Une des techniques a pour but de traduire une sous-classe des annotations *ACSL*– celles dites exécutables – en instructions C intégrées au programme sous analyse [1]. Cette transformation, codée au sein de l'outil *E-ACSL* [2] de *Frama-C*, permet d'obtenir un nouveau programme C dont la correction vis-à-vis de sa spécification est vérifiée dynamiquement, pendant son exécution : cette technique est appelée le *runtime assertion checking*.

Une des difficultés principales de cette transformation réside dans la prise en compte du modèle mémoire du langage C afin d'être en mesure de traduire correctement, par exemple, le prédicat *ACSL* `\valid(p)` qui permet de spécifier que le pointeur `p` est valide (*i.e.* qu'on peut accéder à son contenu en le déréférençant *via* `*p`). Ainsi, un accès à un tableau hors limites (e.g. avec un indice trop grand), ou à une zone mémoire allouée dynamiquement et ensuite libérée, serait invalide. Il est de même possible de traduire le prédicat `\initialized(&x)` pour vérifier pendant l'exécution que la variable `x` a été correctement initialisée (par exemple, avant de lire son contenu).

Pour traduire de tels prédicats en C, *E-ACSL* instrumente notamment le programme initial pour collecter ses allocations, dé-allocations et initialisations *via* des appels de fonctions vers une bibliothèque C dédiée préalablement développée [3].

Néanmoins, cette instrumentation est insuffisante pour détecter correctement tous les cas de validité et d'invalidité de pointeurs. Il reste par exemple des problèmes liés aux structures [1] ou à des initialisations indirectes de zones mémoires dont les incorrections sont difficiles à déterminer.

Le but du stage est d'étendre le schéma de compilation actuel d'*E-ACSL* afin de vérifier correctement des propriétés mémoires sur des programmes C pour lesquelles *E-ACSL* est aujourd'hui incorrect. Cette nouvelle traduction devra ensuite être implémentée en *OCaml* au sein de l'outil *E-ACSL*. Enfin, le code développé sera évalué sur des exemples spécifiques et des *benchmarks* standards pour déterminer la correction et l'efficacité du code généré par rapport à la version actuelle d'*E-ACSL* et à d'autres outils capables de vérifier des propriétés mémoires à l'exécution (comme *AddressSanitizer* de Google ou *MemCheck* de Valgrind).

Références

[1] Mickaël Delahaye, Nikolai Kosmatov, and Julien Signoles. Common specification language for static and dynamic analysis of C programs. In *Symposium on Applied Computing (SAC'13)*, 2013.

1. Voir par exemple les rapports de bogues `https://bts.frama-c.com/view.php?id=2310` et `https://bts.frama-c.com/view.php?id=2327`.

[2] Julien Signoles, Nikolai Kosmatov, and Kostyantyn Vorobyov. E-ACSL, a Runtime Verification Tool for Safety and Security of C Programs. Tool Paper. In *Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CuBES)*, 2017.

[3] Kostyantyn Vorobyov, Julien Signoles, and Nikolai Kosmatov. Shadow state encoding for efficient monitoring of block-level properties. In *International Symposium on Memory Management (ISMM'17)*, June 2017.

Candidatures

Une bonne maîtrise des langages C et *OCaml* et des connaissances en compilation sont nécessaires pour ce stage.

Encadrants : Julien Signoles, Fonenantsoa Maurica et Dara Ly
Contact : Julien Signoles (`prenom.nom@cea.fr`)

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