Proposition de stage niveau bac+5

Synthèse de fonctions C à partir de définitions axiomatiques

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. Le stage se déroulera au sein de l'équipe de R&D développant *Frama-C*.

Objectifs

Chaque programme C analysé par *Frama-C* peut être annoté par des spécifications formelles écrites dans un langage appelé *ACSL* [1]. *Frama-C* [3] offre alors différentes techniques de vérification pour garantir que le programme satisfait sa spécification. Chaque technique de vérification est codée dans un greffon de *Frama-C* dédié. Une de ces techniques a notamment pour but de traduire une sous-classe des annotations *ACSL*– celles dites exécutables et appartenant au langage *E-ACSL* – en instructions C intégrées au programme sous analyse [2]. Cette transformation, programmée dans *Frama-C* dans un greffon appelé (aussi) *E-ACSL* [4], 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 la vérification à l'exécution (*runtime assertion checking*).

Le but du stage est d'élargir la classe des annotations exprimables en *E-ACSL* afin d'y inclure des fonctions et des prédicats logiques qui ne sont pas explicitement définis mais qui sont seulement axiomatisés : leurs comportements sont uniquement définis par un ensemble d'axiomes logiques. Pour ce faire, on pourra s'appuyer sur une traduction similaire convertissant un sous-ensemble des inductifs de *Coq* en fonctions récursives *OCaml* [5].

Plus précisément, le stagiaire devra :

— étudier l'état de l'art, en particulier les travaux de P.-N. Tollitte [5] ;
— définir un sous-ensemble $\mathcal{D}$ syntaxique exécutable des définitions axiomatiques *ACSL* ;
— définir un schéma de compilation des définitions de $\mathcal{D}$ en fonctions récursives *C* en explicitant les hypothèses sous lesquelles la traduction est correcte ;
— programmer cette traduction en *OCaml* en l'intégrant au greffon *E-ACSL* de *Frama-C* ;
— expérimenter le prototype réalisé sur un ensemble de cas représentatifs.

Candidatures

Le candidat idéal aura suivi un cours de logigue et de compilation, maîtrisera un langage de spécification formelle, ainsi que les langages de programmes *C* et *OCaml*.

Contacts : Julien Signoles (`julien.signoles@cea.fr`) et Catherine Dubois (`dubois@ensiie.fr`)

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

Références

[1] P. Baudin, J.-C. Filliâtre, C. Marché, B. Monate, Y. Moy, and V. Prevosto. *ACSL : ANSI/ISO C Specification Language, version 1.7*, 2013. `http://frama-c.com/acsl.html`.

[2] M. Delahaye, N. Kosmatov, and J. Signoles. Common specification language for static and dynamic analysis of C programs. In *Symposium on Applied Computing (SAC'13)*, 2013.

[3] F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, and B. Yakobowski. Frama-C : A Software Analysis Perspective. *Formal Aspects of Computing*, January 2015.

[4] J. Signoles, N. Kosmatov, and K. 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.

[5] P.-N. Tollitte, D. Delahaye, and C. Dubois. Producing certified functional code from inductive specifications. In *Certified Programs and Proofs (CPP'12)*, December 2012.