Postdoctoral Position

Function Synthesis for C Programs from Formal Specifications

Keywords: function synthesis, code generation, formal specification, *Frama-C*, *C*, *ACSL*, *OCaml*.

Context: CEA LIST, Software Security Labs

The Software Security Laboratory (LSL) has an ambitious goal: help designers, developers and validation experts ship high-confidence systems and software. Objects in our surroundings are getting more and more complex, and we have built a reputation for efficiently using formal reasoning to demonstrate their trustworthiness. Within the CEA LIST Institute, LSL is dedicated to inventing the best possible means to conduct formal verification. We design methods and tools that leverage innovative approaches to ensure that real-world systems can comply with the highest safety and security standards. And in doing so, we get to interact with the most creative people in academia and the industry.

Our organizational structure is simple: those who pioneer new concepts are the ones who get to implement them. We are a fast-growing thirty-person team, and your work will have a direct and visible impact on the state of formal verification.

CEA LIST's new offices are located at the heart of Campus Paris Saclay, in the largest European cluster of public and private research.

Work Description

LSL is the main developer of *Frama-C* [4] (`http://frama-c.com`), a code analysis platform for C programs which provides several collaborative analyzers as plug-ins. *Frama-C* itself is developed in *OCaml*. *Frama-C* allows the user to annotate *C* programs with formal specifications written in the *ACSL* specification language [1]. *Frama-C* can then ensure that a *C* program satisfies its formal specification by relying on weakest preconditions calculus or abstract interpretation.

The above-mentionned static analysis techniques may be used on a partial program, that is a program containing external library functions, as soon as they are specified enough. However dynamic analysis techniques like monitoring or testing cannot be used on such partial programs because they cannot be executed. For instance, code generated from *E-ACSL* [3], a *Frama-C* plug-in for checking *ACSL* annotations at runtime, cannot be executed in such a case.

The aim of this postdoc is to address this limitation by designing a way to automatically generate a *C* function body b from an *ACSL* function contract c in a way that b satisfies c. Such a code generation is named function synthesis [5] in the litterature. Like any interesting research problem in formal methods, it is undecidable in the general case.

A first step will be to survey the existing litterature to study what already exists, how to adapt existing works to *C* programs annotated with *ACSL* annotations and how to improve state-of-the-art techniques to handle new kinds of annotations. Then a full compilation scheme will be designed and proved correct. Finally it will be implemented in *OCaml* as a new *Frama-C* plug-in and experimented on a realistic case study.

References

[1] Patrick Baudin, Jean-Christophe Filliâtre, Claude Marché, Benjamin Monate, Yannick Moy, and Virgile Prevosto. *ACSL: ANSI/ISO C Specification Language. Version 1.8*, March 2014.

[2] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C, A software Analysis Perspective. In *Software Engineering and Formal Methods (SEFM)*, October 2012.

[3] Mickaël Delahaye, Nikolai Kosmatov, and Julien Signoles. Common specification language for static and dynamic analysis of C programs. In *the 28th Annual ACM Symposium on Applied Computing (SAC)*, pages 1230–1235. ACM, March 2013.

[4] Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-c: A software analysis perspective. *Formal Aspects of Computing*, pages 1–37, 2015. Extended version of [2].

[5] Viktor Kuncak, Mikaël Mayer, Ruzica Piskac, and Philippe Suter. Complete functional synthesis. In *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '10, pages 316–329, New York, NY, USA, 2010. ACM.

Applications

Knowledge in several of the following fields is required:

- (first order) logic
- semantics of programming languages (in particular, the ISO *C* 99 programming language)
- compilation techniques
- formal specification
- functionnal programming (in particular, *OCaml* programming)

Contact: Julien Signoles (*julien.signoles*@cea.fr)