18-month Postdoc Position

Improving E-ACSL, a Runtime Verification Tool for C Programs

Keywords: runtime verification, program transformation, compilation, static analysis
Programming Languages: *OCaml*, *C*
Tools: *Frama-C*, *E-ACSL*

Context: CEA LIST, Software Security Lab

The Software Reliability Laboratory (LSL) at CEA LIST 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. 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 fourty-person team, and your work will have a direct and visible impact on the state of formal verification. CEA LIST's offices are located at the heart of Campus Paris Saclay, in the largest European cluster of public and private research.

Work Description

Our team develops *Frama-C* [5] (`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 several techniques including abstract interpretation, weakest preconditions calculus, and runtime verification.

E-ACSL is the *Frama-C* plug-in dedicated to runtime verification. It converts a *C* program extended with formal annotations written in a subset of *ACSL* into a new *C* program which checks the validity of annotations at runtime: the program execution stops whenever one annotation is violated, or behaves in the same way than the input program if all its annotations are valid [6]. One key feature of *E-ACSL* is the expressivity of its specification language [2] which allows the user to describe powerful safety and security properties. Another key feature is the efficiency of the generated code which relies on a custom memory library [8] and dedicated static analyses [3, 4].

However *E-ACSL* is still a research prototype and lots of improvements can be done both to extend its expressivity and to improve its efficiency. The goal of the hired postdoc will be to transform this promising research prototype to a powerful runtime verification tool usable to efficiently verify industrial-size *C* programs. To reach this goal, (s)he will have to:

- support the missing features of the specification language (including but not limited to dataflow dependence, memory separation, behavior completeness and disjointness, and set-theoretic constructs see [7]);

- improve the compilation scheme to reduce the memory footprint of the generated code;
- adapt and implement compilation techniques to optimize the efficiency of the generated code;
- design novel dedicated static analyses to minimize the generated code;
- redesign the software architecture of the tool to support the above-mentionned extensions;
- evaluate the improvements against real world programs, either open source or provided by industrial partners.

References

[1] P. Baudin, J.-C. Filliâtre, C. Marché, B. Monate, Y. Moy, and V. Prevosto. *ACSL: ANSI/ISO C Specification Language.*

[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)*, March 2013.

[3] A. Jakobsson, N. Kosmatov, and J. Signoles. Rester statique pour devenir plus rapide, plus précis et plus mince. In *Journées Francophones des Langages Applicatifs (JFLA'15)*, January 2015. In French.

[4] A. Jakobsson, N. Kosmatov, and J. Signoles. Fast as a Shadow, Expressive as a Tree: Optimized Memory Monitoring for C. *Science of Computer Programming*, October 2016.

[5] F. Kirchner, N. Kosmatov, V. Prevosto, J. Signoles, and B. Yakobowski. Frama-c: A software analysis perspective. *Formal Aspects of Computing*, 2015.

[6] N. Kosmatov and J. Signoles. A lesson on runtime assertion checking with Frama-C. In *International Conference on Runtime Verification (RV'13)*, September 2013.

[7] J. Signoles. *E-ACSL Version 1.12.* `http://frama-c.com/download/e-acsl/e-acsl-implementation.pdf`.

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

Application

Knowledge in at least one of the following fields is required:

- *OCaml* programming (at least, functional programming)
- *C* programming
- runtime verification
- compilation
- static analysis
- semantics of programming languages (in particular, the ISO *C99* programming language)
- formal specification

Salary: academic competitive (vary *w.r.t.* diploma and former experience)

Availability: as soon as possible; at least a 3-month procedure for administrative and security purposes

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

Please join a detailed CV, a motivation letter and possibly reference letters.