18-month Postdoc Position

Advanced Runtime Assertion Checking of C Programs

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

Context: CEA LIST, Software Security and Reliability Lab

The Software Security and 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. In collaboration with the most creative people in academia and the industry, we design methods and tools that leverage innovative approaches to ensure that real-world systems can comply with the highest safety and security standards

Our organizational structure is simple: those who pioneer new concepts are the ones who get to implement them. We are a forty-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 [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 several techniques including abstract interpretation, weakest preconditions calculus, and runtime assertion checking.

E-ACSL is the Frama-C plug-in dedicated to runtime assertion checking [10]. 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: it behaves in the same way than the input program if all its annotations are valid, or (by default) the program execution stops whenever one annotation is violated. One key feature of E-ACSL is the expressiveness 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 [12] and dedicated static analyses [3, 8].

Still, many challenging research questions are still opened to go beyond the state of the art of runtime assertion checkers and improve E-ACSL significantly. They include (but are not limited to):

- runtime assertion checking of axiomatic definitions by relying on synthesis techniques [6, 11]
- runtime assertion checking of localized properties that refer to several program points [5, 9]
- runtime assertion checking of frame conditions [7] and data dependency properties
- runtime assertion checking of properties over real numbers

- static analysis for monitor optimisation

In the context of the H2020 European project ENSURESEC (2020-2022) that aims at protecting e-commerce by monitoring, the hired postdoc researcher will collaborate with other engineers and researchers at LSL and possibly in other labs, in order to address several of these challenges. She or he will design, formalize and implement innovative solutions, and prove their soundness.

Application

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

- program development (OCaml, C)
- semantics of programming languages
- compilation
- runtime verification
- static analysis
- formal methods for program verification

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

Availability: 2^d semester 2020; a 3-month procedure for administrative and security purposes is required

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

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)*, 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)*, 2015. In French.

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

[5] P. Kosiuczenko. An Abstract Machine for the Old Value Retrieval. In *International Conference on Mathematics of Program Construction (MPC'10)*, June 2010.

[6] V. Kuncak, M. Mayer, R. Piskac, and P. Suter. Complete Functional Synthesis. In *International Conference on Programming, Language Design and Implementation (PLDI'10)*, June 2010.

[7] H. Lehner. *A Formal Definition of JML in Coq and its Application to Runtime Assertion Checking.* PhD thesis, ETH Zurich, 2011.

[8] D. Ly, N. Kosmatov, F. Loulergue, and J. Signoles. Soundness of a dataflow analysis for memory monitoring. In *Workshop on Languages and Tools for Ensuring Cyber-Resilience in Critical Software-Intensive Systems (HILT'18)*, November 2018.

[9] G. Petiot, B. Botella, J. Julliand, N. Kosmatov, and J. Signoles. Instrumentation of annotated C programs for test generation. In *International Conference on Source Code Analysis and Manipulation (SCAM'14)*, September 2014.

[10] J. Signoles, N. Kosmatov, and K. Vorobyov. E-ACSL, a Runtime Verification Tool for Safety and Security of C Programs. Tool Paper. In *International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CuBES'17)*, 2017.

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

[12] 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)*, 2017.