2-year Postdoc Position

Designing Compilation Techniques for Improving Efficiency of E-ACSL, a Runtime Assertion Checker for C Programs

Keywords: Runtime Assertion Checking, Compilation, Source Code Generation, Static Analysis
Programming Languages: *OCaml*, *C*
Tools: *Frama-C*, *E-ACSL*

Context: CEA LIST, Software Safety and Security Lab

CEA LIST's offices are located at the heart of Université Paris-Saclay, the largest European cluster of public and private research. Within CEA LIST, the Software Safety and Security Lab has an ambitious goal: help designers, developers and validation experts ship high-confidence systems and software.

Systems in our surroundings are getting more and more complex, and we have built a reputation for efficiently using formal reasoning to demonstrate their trustworthiness through the design of methods and tools 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, worldwide.

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

Work Description

Our team develops *Frama-C* [1] (`http://frama-c.com`), a code analysis platform for C programs which provides several 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 [2]. *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 [5]. 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: by default, 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. One key feature of *E-ACSL* is the expressivity of its specification language 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 [6] and dedicated static analyses [3, 4].

However *E-ACSL* can still be improved in many ways in order to extend its expressivity and to reduce its memory and time overheads. To reach this goal, (s)he shall make several contributions, for example regarding some of the following objectives:

- improve the current compilation scheme to reduce the memory footprint of the generated code;
- design new compilation techniques and/or adapt existing ones in order to optimize the efficiency of the generated code;

- design novel dedicated fast static analyses to minimize the generated code;
- design one or several new intermediate representations that would make these developments easier
- evaluate the benefits of these improvements on concrete benchmarks and/or use cases

References

[1] P. Baudin, F. Bobot, D. Bühler, L. Correnson, F. Kirchner, N. Kosmatov, A. Maroneze, V. Perrelle, V. Prevosto, J. Signoles, and N. Williams. The Dogged Pursuit of Bug-Free C Programs: The Frama-C Software Analysis Platform. *Communications of the ACM*, 2021.

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

[3] N. Kosmatov, F. Maurica, and J. Signoles. Efficient Runtime Assertion Checking for Properties over Mathematical Numbers. In *International Conference on Runtime Verification (RV)*, October 2020.

[4] 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)*, November 2018.

[5] 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)*, 2017.

[6] K. Vorobyov, J. Signoles, and N. Kosmatov. Shadow State Encoding for Efficient Monitoring of Block-level Properties. In *International Symposium on Memory Management (ISMM)*, 2017.

Application

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

- *OCaml* programming (at least, functional programming)
- *C* programming
- compilation
- semantics of programming languages
- static analysis
- runtime verification
- formal specification

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

Availability: as soon as possible; yet a 3+-month procedure for administrative and security purposes is required

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

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