PhD Position

Outline Runtime Assertion Checking

Keywords: Runtime Assertion Checking, Outline Monitoring, Compilation, Source Code Generation
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 [4]. 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. For doing so, *E-ACSL* performs an heavy implementation of the original source code to insert its own code that monitors the *ACSL* annotations. This technique is usually refered to as *(online) inline runtime verification* [3]. However, depending on the context of application, this heavy instrumentation may lead to prohibitive memory and runtime overheads, as well as security concerns.

The goal of the PhD consists in designing and implementing an *outline runtime verification technique* for *E-ACSL*, compatible with the existing inline technique. Outline runtime verification consists in placing the monitor in an external entity (e.g., another thread, or a remote server) for limiting the instrumentation to communication activities with the remote monitor [3]. While this technique is well known and often applied to monitoring of temporal properties, it was never applied to runtime assertion checking, which raises several challenges regarding the data that need to be monitored. The expected contributions are:

- designing a new instrumentation scheme for outline runtime assertion checking;

- designing a new internal representation for the monitor, usable for both inline and outline runtime assertion checking;
- implementing the new representation and the new instrumentation scheme inside *E-ACSL*;
- evaluating the new outline runtime assertion checker on representative benchmarks and/or use cases;
- (optionally,) designing and implementing optimization strategies to reduce the time and/or memory overheads

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] Yliès Falcone, Klaus Havelund, and Giles Reger. A tutorial on runtime verification. In *Engineering Dependable Software Systems.* IOS Press, 2013.

[4] 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.

Application

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

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

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, and a motivation letter.