Title: Revisiting Constraint Acquisition Through the Lens of Program Specification Synthesis

Supervisors: Sébastien Bardin (CEA Paris - sebastien.bardin@cea.fr), Nadjib Lazaar (LIRMM, Montpellier – lazaar@lirmm.fr), Arnaud Gotlieb (Simula, Oslo – arnaud@simula.no)

Context.
Formal methods have made tremendous progress over the past 20 years, enabling the design and verification of very high-quality software systems in key areas such as aeronautics, compilers, drivers and several parts of operating systems. Such advanced program analysis techniques ultimately rely on *deductive methods* (typically, constraint solvers), i.e. powerful automated reasoning engines able to check the validity of some logical constraint languages, or find counter-examples to these constraints. Yet, a big problem for the adoption of formal methods is that often the user must provide some form of (likely-)invariant or property to be checked. In other terms, such program analyzers can prove facts given by a user, but they cannot infer those facts.

On the other hand, machine learning approaches can indeed infer facts from data (generalization), and thus we can imagine inferring unproven properties or likely-invariants from source code, runtime information, documentation, etc. Yet, such inductive approach comes with no guarantee.

Our overall goal is to understand how deductive methods from program analysis can be combined together with inductive methods from artificial intelligence, so that inferred likely-invariants could be proven or refined, until either proven invariants or definite counter-examples are found.

Potential applications in cybersecurity include automatically finding invariants of a piece of code, as well as automatic code hardening or automatic simplification of malware protections.

Goal & Challenges.

The general goal of this PhD work is to understand how deduction-based approaches (from formal methods and code analysis) and learning-based approaches (from AI and optimization) can be combined together for attacking hard challenges arising from security-oriented program analysis.

We will especially focus on Constraint Acquisition [3], as symbolic learning is closer

to the logical formalism of deductive methods (e.g., symbolic reasoning [1,2]) used in program analysis.

In recent work [4], we have shown some interesting links between Constraint Acquisition and automated software contracts inference, showing that indeed CA can be adapted in a fruitful manner to this problem, allowing to overcome existing methods from the formal verification community. Interestingly, using CA in program analysis allows us to use program executions to answer queries in order to avoid the ad-hoc intervention of human oracle, removing one of the main bottlenecks of CA methods.

How To Apply.

Applicants should contact via email Sébastien Bardin (sebastien.bardin@cea.fr), Nadjib Lazaar (lazaar@lirmm.fr), Arnaud Gotlieb (arnaud@simula.no) with:

- A full curriculum vitae, including a summary of previous research experience.

- A transcript of higher education records.

- A one-page research statement discussing how the candidate's background fits the proposed topic.

- Two support letters of persons that have worked with them.

References.

[1] Cristian Cadar, Koushik Sen: Symbolic execution for software testing: three decades later. Commun. ACM 56(2): 82-90 (2013).

[2] Clark Barrett, Cesare Tinelli: Satisfiability Modulo Theories. Handbook of Model Checking 2018: 305-343.

[3] Christian Bessiere, Frédéric Koriche, Nadjib Lazaar, Barry O'Sullivan: Constraint acquisition. Artif. Intell. 244: 315-342 (2017).

[4] Grégoire Manguy, Sébastien Bardin, Nadjib Lazaar, Arnaud Gotlieb: Automated Program Analysis: Revisiting Precondition Inference through Constraint Acquisition. IJCAI-ECAI 2022.