COCOA: Checking OCaml-C FFI Code by Abstract Interpretation

M2 internship proposal for Spring 2027

Raphaël Monat (Inria Lille) Benoît Montagu (Inria Rennes)
Informal enquiries are welcome by email (in English or French)
Topic can be adjusted, do feel free to reach out.

1 Context

OCaml is a safe programming language: by design of its syntax, semantics, and static type system, the typical runtime errors that can occur in other programming languages, such as buffer overflows, are impossible. However, through the OCaml foreign function interface (FFI), some OCaml programs call C code, either in order to access system libraries, or to achieve greater performance. Contrary to OCaml, C has unsafe, manual memory management. As a result, the combination of OCaml and C is unsafe.

The abstract interpretation framework created by Radhia and Patrick Cousot [2] has been particularly successful to certify the absence of such runtime errors in critical embedded C software [3]. This framework has also been used in multiple other contexts, including for the analysis of OCaml programs [1, 6, 7, 10, 11], or for multilanguage analyses for Python+C programs [8].

2 Goal

The overall, long-term goal of the COCOA project is to deploy a safety analyzer for the OCaml-C FFI code. This analyzer would regularly analyze all opam packages and provide actionable reports to developers.

This internship acts as a starting point of this project. It will focus on a restricted number of OCaml-C FFI features, such as ones related to the garbage collector (root declaration) and the runtime lock. The internship will start with a study of related work, both on the semantics of the OCaml-C FFI [5], and on previous multilanguage OCaml-C analyses [4, 9]. One particular concern is to ensure that the safety obligations are coherent with the assumptions performed by the OCaml runtime.

3 Developed Skills

The candidate will work on developing conservative static analyses, and in particular some rooted in the framework of abstract interpretation. We expect the successful candidate to be motivated to improve experimental research tools such as Mopsa, which is implemented in the OCaml functional programming language. Communication skills are a plus, and will be further developed during the internship, both through text reports and presentations.

4 Logistics

The intern will be part of the EPICURE team of Inria Rennes, or the SyCoMoRES team of Inria Lille. We plan to hold weekly research meetings. We have funding to cover an internship. If the internship goes well for all involved parties, we plan to follow up with a PhD grant. Funding is currently partially acquired for this PhD, that would take place in Rennes.

Bibliography

[1] Santiago Bautista, Thomas P. Jensen, and Benoît Montagu. “An input-output relational domain for algebraic data types and functional arrays”. In: *Formal Methods Syst. Des.* 1 (2025). DOI: 10.1007/S10703-024-00456-Z.

[2] Patrick Cousot and Radhia Cousot. “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. In: *POPL 1977*. 1977. DOI: 10.1145/512950.512973.

[3] Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. “Combination of Abstractions in the ASTRÉE Static Analyzer”. In: *ASIAN 2006*. 2006. DOI: 10.1007/978-3-540-77505-8_23.

[4] Michael Furr and Jeffrey S. Foster. “Checking type safety of foreign function calls”. In: *ACM Trans. Program. Lang. Syst.* 4 (2008). DOI: 10.1145/1377492.1377493.

[5] Armaël Guéneau, Johannes Hostert, Simon Spies, Michael Sammler, Lars Birkedal, and Derek Dreyer. “Melocoton: A Program Logic for Verified Interoperability Between OCaml and C”. In: *Proc. ACM Program. Lang.* OOPSLA2 (2023). DOI: 10.1145/3622823.

[6] Pierre Lermusiaux and Benoît Montagu. “Detection of Uncaught Exceptions in Functional Programs by Abstract Interpretation”. In: *Programming Languages and Systems - 33rd European Symposium on Programming, ESOP 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part II*. 2024. DOI: 10.1007/978-3-031-57267-8_15.

[7] Pierre Lermusiaux and Benoît Montagu. “The Salto Project: Static Analysis of OCaml Programs by Abstract Interpretation”. In: *ERCIM News* 139 (2024). URL: https://ercim-news.ercim.eu/en139/special/the-salto-project-static-analysis-of-ocaml-programs-by-abstract-interpretation.

[8] Raphaël Monat, Abdelraouf Ouadjaout, and Antoine Miné. “A Multilanguage Static Analysis of Python Programs with Native C Extensions”. In: *Static Analysis - 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17-19, 2021, Proceedings*. 2021. DOI: 10.1007/978-3-030-88806-0_16.

[9] Edwin Török. “Targeted Static Analysis for OCaml C Stubs: eliminating gremlins from the code”. In: *CoRR* (2023). DOI: 10.48550/ARXIV.2307.14909. arXiv: 2307.14909.

[10] Milla Valnet, Raphaël Monat, and Antoine Miné. “Compositional Static Value Analysis for Higher-Order Numerical Programs”. In: *39th European Conference on Object-Oriented Programming, ECOOP 2025, Bergen, Norway, June 30 - July 2, 2025*. 2025. DOI: 10.4230/LIPICS.ECOOP.2025.32.

[11] Milla Valnet, Raphaël Monat, and Antoine Miné. “DelExp: A Relational Container Abstraction: with Applications to Compositional Analysis”. In: *40th European Conference on Object-Oriented Programming, ECOOP 2026, Brussels, Belgium, June 29 - July 3, 2026*. 2026. DOI: 10.4230/LIPICS.ECOOP.2026.29.