Internship Proposal:
Infinite Structures for Program Verification

Advisors

Théo Laurent
Team Prosecco – Inria
theo.laurent@inria.fr

Aymeric Fromherz
Team Prosecco – Inria
aymeric.fromherz@inria.fr

1 Context

Verification in F⋆ The F⋆ programming language [6, 5] is a general purpose language designed with formal verification in mind. It features programming with side-effects and dependent types, and proving via SMT-based reasoning as well as tactics. It has proven its strength with the implementation of several large scale project such as HACL⋆ [8], a verified cryptographic library featured in Firefox and WireGuard.

Infinite Structures with Coinduction Coinduction is a technique used to define and manipulate infinite data structures. It is particularly useful in program verification when doing proof by simulation. For instance, it is used in the verified C compiler CompCert [4], to model execution traces. In general, coinduction is useful to model potentially non-terminating programs and their interaction with their environment, see for example the Interaction Trees [7] library. While it has proven its value as a proof technique, coinduction is not yet supported in F⋆.

2 Goal

The goal of this internship is to provide support for coinduction in F⋆. We aim at tackling two challenges:

- The first objective is to provide an interface usable for defining and using coinductive data structures, as well as proving facts by coinduction.
- The second objective is to back up this implementation with solid formal foundations.

To solve those challenges, we will leverage the F⋆ metaprogramming features. The implementation should be suitable for realistic developments relying on coinduction. In particular, having in mind the F⋆ ecosystem, it should feature good interaction with the SMT solver. To this end, we will draw inspiration from existing work in the Dafny verification system [3]. On the formalization side, we will look at previous work on encoding coinductive structures with regular inductive types [1]. The approach we suggest is to show that the interface of the library is faithful with regard to a correct-by-construction encoding of coinductive types. The implementation of the library will be showcased with concrete use cases, for example one could port the aforementioned Interaction Trees library to F⋆ and use it to prove properties about programs with arbitrary recursion and side effects.

As an additional objective, depending on the progression of the internship, we are also interested in adapting techniques from the Paco [2] library for the Coq Proof Assistant. This library provide an original and effective way do prove facts coinductively. It will help us provide a tactic based approach to coinductive proofs when the SMT approach is too coarse-grained.

3 Qualifications

The internship will be six month and is targeted at master degree students, it will be hosted at Inria Paris by the Prosecco team. The preferred qualification for the candidate are familiarity with type theory and functional programming. Experience with a proof assistant is a plus.

References

[1] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Containers: Constructing strictly positive types. *Theor. Comput. Sci.*, 342(1):3–27, sep 2005.

[2] Chung-Kil Hur, Georg Neis, Derek Dreyer, and Viktor Vafeiadis. The power of parameterization in coinductive proof. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, page 193–206, New York, NY, USA, 2013. Association for Computing Machinery.

[3] K. Rustan M. Leino and Michał Moskal. Co-induction simply. In Cliff Jones, Pekka Pihlajasaari, and Jun Sun, editors, *FM 2014: Formal Methods*, pages 382–398, Cham, 2014. Springer International Publishing.

[4] Xavier Leroy. A formally verified compiler back-end. *Journal of Automated Reasoning*, 43(4):363–446, 2009.

[5] Guido Martínez, Danel Ahman, Victor Dumitrescu, Nick Giannarakis, Chris Hawblitzel, Cătălin Hriţcu, Monal Narasimhamurthy, Zoe Paraskevopoulou, Clément Pit-Claudel, Jonathan Protzenko, Tahina Ramananandro, Aseem Rastogi, and Nikhil Swamy. Meta-F*: Proof automation with SMT, tactics, and metaprograms. In *28th European Symposium on Programming (ESOP)*, pages 30–59. Springer, april 2019.

[6] Nikhil Swamy, Joel Weinberger, Cole Schlesinger, Juan Chen, and Benjamin Livshits. Verifying higher-order programs with the Dijkstra monad. In *Proceedings of the 34th annual ACM SIGPLAN conference on Programming Language Design and Implementation*, PLDI '13, pages 387–398, 2013.

[7] Li-yao Xia, Yannick Zakowski, Paul He, Chung-Kil Hur, Gregory Malecha, Benjamin C. Pierce, and Steve Zdancewic. Interaction trees: Representing recursive and impure programs in coq. *Proc. ACM Program. Lang.*, 4(POPL), dec 2019.

[8] Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Jonathan Protzenko, and Benjamin Beurdouche. HACL*: A verified modern cryptographic library. In *ACM Conference on Computer and Communications Security*, pages 1789–1806. ACM, 2017.