Machine-checked undecidability proofs regarding contextual equivalence in PCF

Yannick Forster and Guilhem Jaber

Gallinette Project-Team, Inria, Nantes, France

En bref / In short

Laboratoire, institution / Laboratory, institution: Inria, Université de Nantes (LS2N).
Lieu du stage / Location: LS2N, UFR Sciences et Techniques, 2, rue de la Houssinière, Nantes.
Équipe d'accueil / Host team: Équipe-projet Inria Gallinette.
Contact: Yannick Forster and Guilhem Jaber (e-mail)
Encadrement / Supervision: The internship will take place in the Inria Gallinette team. It will be co-supervised by Yannick Forster and Guilhem Jaber.
Mots clefs / Keywords: Computability theory, programming languages theory, undecidability, contextual equivalence, type theory, Coq
Langue de travail / Working language: English.

Description

This project aims at giving machine-checked undecidability proofs of undecidability of various problems regarding contextual equivalence in PCF in the proof assistant Coq [10]. The final goal is to give a (possibly simplified) undecidability proof of contextual equivalence of finitary PCF, a famously hard result in the area of semantics of programming languages due to Loader [6].

Context

In semantics of programming languages, a key aspect is to determine when the denotations of two programs in a model are equal. The gold standard of denotational semantics if to provide *fully abstract models*, where the equality of denotation coincides with contextual equivalence, providing the maximal adequate equational theory. Since its introduction by Plotkin [9], it was a long-standing open problem whether a fully abstract semantics can be given for PCF, a paradigmatic functional programming language. It is a simply-typed λ-calculus with natural numbers and a fixed-point combinator.

It is easy to see that contextual equivalence in PCF is undecidable, because PCF is Turing-complete. As a consequence, there is no computable, fully-abstract model of PCF – such a model would give rise to a decision procedure of contextual equivalence. Restricted to finitary PCF, the fragment of PCF with booleans and a constant $\bot$ representing non-termination, it was expected that a model would be effectively computable. This means that a fully abstract model would provide a decision procedure for contextual equivalence for this fragment. However, Ralph Loader proved that contextual equivalence for finitary PCF is undecidable [6], closing negatively the quest for a fully abstract model for PCF. The undecidability proof is famously hard and technical, and not presented in text books on semantics or computability. It is by reduction from the word problem of first-order string rewriting systems (so-called semi-Thue systems).

Coq's type theory is the ideal ground to give machine-checked undecidability proofs: The Coq library of undecidability proofs [4] provides a library of problems which are all shown undecidable and can be used as starting point for undecidability proofs by reduction. The proofs in the library rely on a so-called synthetic approach to undecidability [3], based on the fact that all functions definable in constructive type theory in general and Coq in particular are computable. Since we know that the halting problem of Turing machines cannot have a Coq decision function (because such a function is not computable), any problem the halting problem many-one reduces to can also not have one. This insight is used as the basis of a definition of undecidability which does not have to rely on a formal model of computation, thereby circumventing tedious formalisation of such models.

Objectives

The aim of this project is to give a Coq proof of the undecidability results for PCF and finitary PCF, possibly with some intermediate results to test strategies or simplify the final proof.

For PCF there exist various undecidability proofs of contextual equivalence, all intuitively amounting to proving that PCF is Turing complete. Concretely, it is necessary to give a many-one reduction from the halting problem of any Turing-complete language to PCF, or more generally from any undecidable problem. To formalise the undecidability proof in Coq, it is important to first find the simplest possible undecidability proof. Usually, and contrarily to text book proof sketches, simple and fully formal undecidability proof do not start at the halting problem of Turing machines, but rather with computational models such as two counter machines as introduced by Minsky [8, 2], FRACTRAN as introduced by Conway [1], or formulations of solvability of Diophantine equations as in Hilbert's 10th problem [7, 5]. The first challenge of the project is to identify a suitable starting problem for a reduction, formalise it, and give the proof in Coq. The starting problems can be chosen from the Coq library of undecidability proofs.

Second, the same strategy is used for Loader's result that contextual equivalence in finitary PCF is undecidable: Analysing whether a more suitable starting problem than semi-Thue systems is available, formalise the reduction, and give the proof in Coq.

We expect a student to gain new knowledge in undecidability proofs and programming language semantics, and acquire or hone existing skills in the Coq proof assistant. While a familiarity with typed lambda calculi and basic skills in Coq is required, all further knowledge can be acquired during the project. An outstandingly successful completion of the project could potentially be published at a conference like Interactive Theorem Proving (ITP) or Certified Programs and Proofs (CPP).

References

[1] John H Conway. Fractran: A simple universal programming language for arithmetic. In *Open Problems in Communication and Computation*, pages 4–26. Springer, 1987. doi:10.1007/978-1-4612-4808-8_2.

[2] Andrej Dudenhefner. Certified decision procedures for two-counter machines. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSCD.2022.16.

[3] Yannick Forster. *Computability in Constructive Type Theory*. PhD thesis, Saarland University, 2021. doi:10.22028/D291-35758.

[4] Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq library of undecidable problems. In *The Sixth International Workshop on Coq for Programming Languages (CoqPL 2020)*., 2020. URL: https://github.com/uds-psl/coq-library-undecidability.

[5] Dominique Larchey-Wendling and Yannick Forster. Hilbert's tenth problem in coq. 2019. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10534/, doi:10.4230/LIPICS.FSCD.2019.27.

[6] Ralph Loader. Finitary PCF is not decidable. *Theor. Comput. Sci.*, 266(1-2):341–364, 2001. doi:10.1016/S0304-3975(00)00194-8.

[7] Yuri V. Matiyasevich. Martin davis and hilbert's tenth problem. In Eugenio G. Omodeo and Alberto Policriti, editors, *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*, volume 10 of *Outstanding Contributions to Logic*, pages 35–54. Springer, 2016. doi:10.1007/978-3-319-41842-1_2.

[8] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., USA, 1967.

[9] Gordon D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):223–255, 1977. doi:10.1016/0304-3975(77)90044-5.

[10] The Coq Development Team. The Coq proof assistant, September 2022. doi:10.5281/zenodo.7313584.