A Coq Library of Undecidable Problems

Yannick Forster, Saarland University
Dominique Larchey-Wendling, Université de Lorraine, CNRS, LORIA
Andrej Dudenhefner, Saarland University
Edith Heiter, Saarland University
Dominik Kirst, Saarland University
Fabian Kunze, Saarland University
Gert Smolka, Saarland University
Simon Spies, Saarland University, U. of Cambridge
Dominik Wehr, Saarland University, U. van Amsterdam
Maximilian Wuttke, Saarland University

Abstract

We propose a talk on our library of mechanised reductions to establish undecidability results in Coq. The library is a collaborative effort, growing constantly and we are seeking more outside contributors willing to work on undecidability results in Coq.

1 Introduction

Undecidability proofs are usually carried out by giving a reduction from an undecidable problem to the problem to be shown undecidable. It involves the definition of a reduction function, proving that it is both computable and correct (as a reduction). We base our library[1] on a synthetic approach to undecidability [7], meaning we rely on Coq's built-in notion of computation. Since every function on data types (such as $\mathbb{N}$) definable in (axiom free) Coq is computable in any standard model of computation, the computability requirement is automatically fulfilled, and giving a reduction only amounts to defining a Coq term and proving its correctness.

Most undecidability proofs work by many-one reductions. A problem $p : X \to \mathrm{Prop}$ is many-one reducible to $q : Y \to \mathrm{Prop}$, written $p \preceq q$, if $\forall x : X,\ p\,x \leftrightarrow q(f\,x)$ holds for some $f : X \to Y$, the correctness statement meaning that f embeds p-validity into q-validity.[2] Many-one reductions form a sub-class of Turing reductions: a problem p is Turing-reducible to a problem q if a decider for q can be turned into a decider for p, i.e. if the type $\mathrm{dec}\,q \to \mathrm{dec}\,p$ is inhabited, where $\mathrm{dec}\,p := \forall x,\ \{p\,x\} + \{\neg p\,x\}$.

In total, our library contains more than 20 different problems and about 70 000 lines of code. In this abstract, we give an overview of the problems in the library, the reductions between the problems, and sum up possible future work.

2 Problems in the Library

The problems in our library can mostly be categorized into seed problems, advanced problems, and target problems.

As they are simple to state, seeds make for good starting points leading to smooth reductions to other problems.

Advanced problems do not work well as seeds, but they highlight the potential of our library as a framework for mechanically checking pen&paper proofs of potentially hard undecidability results.

Target problems are very expressive and thus work well as targets for reduction, with the aim of closing loops in the reduction graph (Figure 1) to establish the inter-reducibility of problems.

[1]Which can be found at https://github.com/uds-psl/coq-library-undecidability/.

[2]Using dependent types, one can also pack a reduction $p \preceq q$ together with its correctness statement in a dependent pair i.e. $\forall x : X,\ \{y : Y \mid p\,x \leftrightarrow q\,y\}$.

CoqPL '20, January 25, 2020, New Orleans, LA., USA
2020.

2.1 Seeds

- The Post correspondence problem (PCP), which is a matching problems on strings, mechanised in [6]. PCP works well as seed for target problems that can express string concatenation and simple inductive predicates. In particular, PCP on Boolean strings (BPCP) is simpler than most halting problems, and thus is oftentimes used as the starting point of reductions.
- Halting problems for (n or just two) registers machines (MM, MM_0 and MM2), also known as Minsky machines, mechanised in [10]. These machines have a (fixed) number of $\mathbb{N}$-valued registers and can increase/decrement registers or perform conditional jumps. The machines work well as seed for target problems based on simple arithmetic operations e.g. +1 or −1.
- The halting problem for the FRACTRAN language, which is a very simple language where states are natural numbers and programs are lists of fractions, mechanised in [15]. FRACTRAN works well as seed for target problems which can simulate more involved arithmetic, in this case multiplication.
- Satisfiability of Diophantine equations (H10) or elementary Diophantine constraints ($\mathrm{H10_c}$), which are equivalent formulations of Hilbert's tenth problem, mechanised in [15].

2.2 Advanced Problems

- The halting problems for single-tape (TM_s) and multi-tape Turing machines (TM_m), mechanised in [9]. Turing machines are the standard model of computation, but their formal definition is quite involved compared to e.g. PCP.[3]
- The halting problem for binary stack machines (BSM), which are simple machines working with stacks of Booleans, mechanised in [10]. They can push and pop Booleans, and conditionally jump.
- Standard first-order string rewriting (SR), mechanised in [6].
- Entailment in (elementary and standard) intuitionistic linear logic (EILL and ILL), mechanised in [10].
- Provability and satisfiability for intuitionistic and classical first-order logic (FOL), mechanised in [7].
- Type inhabitation for System F ($\Gamma \vdash_{\mathrm{F}} ? : A$), described in [5].
- Finite satisfiability in (classical) first-order logic, known as Trakhtenbrot's theorem (Trakht.)
- The termination problem for μ-recursive algorithms (μ-rec), a standard class of formal algorithms, mechanised in [14].
- The intersection problem (CFI) and palindrome problem (CFP) for context-free grammars, mechanised in [6].
- 2^{nd}-order (2oUnif), 3^{rd}-order (3oUnif) and thus higher-order unification in the simply-typed λ-calculus, mechanised in [20].

[3]Turing machines are often used as seeds in the literature but we found that using alternative seeds makes for simpler mechanised reductions in many instances.

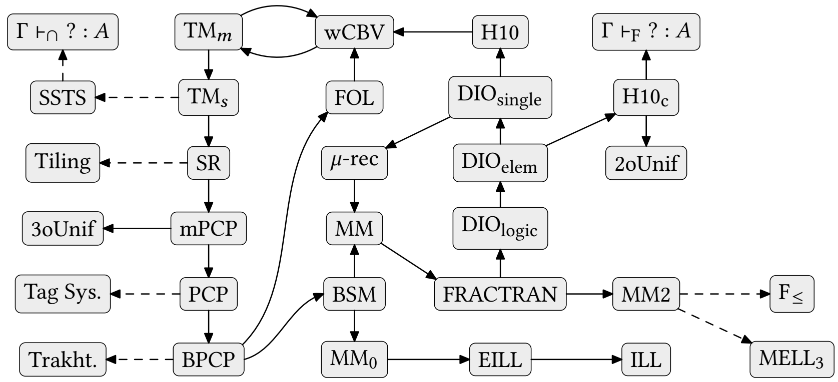

Figure 1. Graph of reductions contained in the library, dashed lines are future work.

2.3 Target Problems

- The halting problem for the weak call-by-value λ-calculus L (wCBV), mechanised in [11]. In order to reduce a problem p to L-halting, one first shows that the problem is enumerable in Coq, i.e. that there is a term $f : \mathbb{N} \to \mathsf{list}\ X$ s.t. $\forall x,\ p\,x \leftrightarrow \exists n,\ x \in f\ n$. As a second step, one can use the automatic translation of Coq terms to L from [8] to immediately obtain a reduction from p to L-halting. This technique is used to obtain a compact reduction from provability in first-order logic to L-halting in [6].
- Provability or satisfiability in first-order logic, mechanised in [7]. If a problem is expressible with a first-order formula, this immediately serves as reduction to provability on first-order logic.

3 Future Work

First, we would like to strengthen the theoretical basis our work is built on, i.e. the folklore fact that every function of type e.g. $\mathbb{N} \to \mathbb{N}$ definable in (axiom free) Coq can be shown computable in a model of computation. A mechanised proof of this could for instance be carried out based on MetaCoq [19]. We also want to analyse which classical axioms are compatible with this computability assumption.

Secondly, we would like to include existing mechanisations of undecidability results in a uniform fashion, for instance [3] or [4].

Thirdly, there are many undecidability proofs we would like to mechanise in the future. Several known undecidability results build on the undecidability of unification problems we already include (2oUnif or 3oUnif), such as typability in the $\lambda\Pi$-calculus [2] or type inference in Curry-style System F_ω [21]. The intersection problem for two-way-automata [18] is shown undecidable by reduction from PCP. Tiling problems are often used in undecidability proofs for logic and shown undecidable by reduction from TM_s. Post's tag systems (Tag Sys.) can be shown undecidable by e.g. reduction from PCP [17]. Concerning linear logic, the reduction from MM2 to MELL_3 (with 3 !-modalities) is a priority [1]. Deductive consequence in axiomatic systems such as Peano arithmetic or ZF set theory can be shown undecidable by reduction from e.g. PCP, H10 or $\mathrm{H10}_\mathrm{c}$.

Subtyping in $\mathrm{F}_\leq$ was shown undecidable in [16], and typability and type checking for System F were shown undecidable in [22]. By including those problems in the library, we might be able to connect the mechanised undecidability of type checking in $D_{<:}$ [12].

Lastly, the undecidability proof of semi-unification [13] is by reduction from the mortality problem of Turing machines. It would be a challenge to find a different starting point for this proof to make it suitable for mechanisation in Coq.

References

[1] Kaustuv Chaudhuri. 2018. Expressing additives using multiplicatives and subexponentials. *Mathematical Structures in Computer Science* 28, 5 (2018), 651–666.
[2] Gilles Dowek. 1993. The undecidability of typability in the lambda-pi-calculus. In *International Conference on Typed Lambda Calculi and Applications*. Springer.
[3] Andrej Dudenhefner and Jakob Rehof. 2017. Rank 3 Inhabitation of Intersection Types Revisited (Extended Version). *arXiv preprint arXiv:1705.06070* (2017).
[4] Andrej Dudenhefner and Jakob Rehof. 2018. Lower End of the Linial-Post Spectrum. In *23rd International Conference on Types for Proofs and Programs (TYPES 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[5] Andrej Dudenhefner and Jakob Rehof. 2019. A Simpler Undecidability Proof for System F Inhabitation. In *24th International Conference on Types for Proofs and Programs (TYPES 2018) (LIPIcs)*, Vol. 130.
[6] Yannick Forster, Edith Heiter, and Gert Smolka. 2018. Verification of PCP-related computational reductions in Coq. In *International Conference on Interactive Theorem Proving*. Springer, 253–269.
[7] Yannick Forster, Dominik Kirst, and Gert Smolka. 2019. On synthetic undecidability in Coq, with an application to the Entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs.*
[8] Yannick Forster and Fabian Kunze. 2019. A Certifying Extraction with Time Bounds from Coq to Call-By-Value Lambda Calculus. In *10th International Conference on Interactive Theorem Proving (LIPIcs)*, Vol. 141. 17:1–17:19.
[9] Yannick Forster, Fabian Kunze, and Maximilian Wuttke. 2020. Verified Programming of Turing Machines in Coq. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs.*
[10] Yannick Forster and Dominique Larchey-Wendling. 2019. Certified undecidability of intuitionistic linear logic via binary stack machines and Minsky machines. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs.* ACM, 104–117.
[11] Yannick Forster and Gert Smolka. 2017. Weak call-by-value lambda calculus as a model of computation in Coq. In *International Conference on Interactive Theorem Proving*. Springer, 189–206.
[12] Jason Hu and Ondřej Lhoták. 2019. Undecidability of $D_{<:}$ and Its Decidable Fragments. *arXiv preprint arXiv:1908.05294* (2019).
[13] Assaf J Kfoury, Jerzy Tiuryn, and Pawel Urzyczyn. 1993. The undecidability of the semi-unification problem. *Information and Computation* 102, 1 (1993).
[14] Dominique Larchey-Wendling. 2017. Typing Total Recursive Functions in Coq. In *International Conference on Interactive Theorem Proving*. Springer, 371–388.
[15] Dominique Larchey-Wendling and Yannick Forster. 2019. Hilbert's Tenth Problem in Coq. In *4th International Conference on Formal Structures for Computation and Deduction (LIPIcs)*, Vol. 131. 27:1–27:20.
[16] Benjamin C Pierce. 1994. Bounded quantification is undecidable. *Information and Computation* 112, 1 (1994), 131–165.
[17] Emil L Post. 1943. Formal reductions of the general combinatorial decision problem. *American journal of mathematics* 65, 2 (1943), 197–215.
[18] Michael O. Rabin and Dana Scott. 1959. Finite automata and their decision problems. *IBM journal of research and development* 3, 2 (1959), 114–125.
[19] Matthieu Sozeau, Abhishek Anand, Simon Boulier, Cyril Cohen, Yannick Forster, Fabian Kunze, Gregory Malecha, Nicolas Tabareau, and Théo Winterhalter. 2019. The MetaCoq Project. (June 2019). https://hal.inria.fr/hal-02167423
[20] Simon Spies and Yannick Forster. 2020. Undecidability of Higher-Order Unification Formalised in Coq. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs.*
[21] Pawel Urzyczyn. 1997. Type Reconstruction in F_ω. *Mathematical. Structures in Comp. Sci.* 7, 4 (Aug. 1997), 329–358. https://doi.org/10.1017/S0960129597002302
[22] Joe B Wells. 1999. Typability and type checking in System F are equivalent and undecidable. *Annals of Pure and Applied Logic* 98, 1-3 (1999), 111–156.