Mechanising Undecidability results in Coq: Elementary Linear Logic and Boolean BI

Dominique Larchey-Wendling, LORIA – CNRS

December 15, 2017

Abstract

We present a constructive Coq mechanization of undecidability results for linear logic (more precisely the elementary fragment) and as a consequence Boolean BI.

The mechanisation of undecidability results in proof assistants in general (and in Coq in particular) is a difficult task because the mechanisation of computability theory has only started recently [7, 8, 4, 2]. Undecidability results are often established by reduction to other previously known undecidability results. Typically for linear logic, by reduction to the computation of counter or Minsky machines [6, 3, 5].

Hence, before showing the undecidability of elementary linear logic, one has to show that the computation of Minsky machines is undecidable. As a side remark, it seems that some people are still unconvinced of the Turing completeness of Minsky machines.[1]

To fully mechanise such undecidability results in proof assistants, we cannot assume a library of already established undecidability results because the developement of such a library is only at its very earlier stages. Hence, for the case of linear logic, we need a mechanised reduction of Turing machines (TM) to Minsky machines (MM). Direct certified programming with TM or MM is very difficult. Actually, it is never done in pen and paper proofs. Even reductions of MM problems to logic problems (or other problems), are rarely or never proved computable in papers. The reason being that doing so would involve very detailed description of encodings with corresponding correctedness proofs. These might obfuscate the "interesting" aspects of reductions. But mechanisation requires a much higher level of details.

Instead of certified programming in assembly like languages (TM or MM), we propose a certified compiler of high-level languages (functional or weak call-by-value λ-calculus [2]) to TM/MM. Hence, we do not need to show the correction of MM simulating TMs but only the correction of a TM interpreter written in a high-level language. This is still difficult but much more doable than proving the correction of an interpreter written in assembly language.

Then we can mecanize our reduction [5] of MM to elementary linear logic and get of certified proof of undecidability of linear logic and Boolean BI, the elementary fragment being common to these two logics.

[1] `https://sites.ualberta.ca/~bimbo`

References

[1] Mauricio Ayala-Rincón and César A. Muñoz, editors. *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, volume 10499 of *Lecture Notes in Computer Science*. Springer, 2017.

[2] Yannick Forster and Gert Smolka. Weak call-by-value lambda calculus as a model of computation in coq. In Ayala-Rincón and Muñoz [1], pages 189–206.

[3] Max Kanovich. Linear Logic as a Logic of Computations. *Annals of Pure and Applied Logic*, 67(1–3):183–212, 1994.

[4] Dominique Larchey-Wendling. Typing total recursive functions in coq. In Ayala-Rincón and Muñoz [1], pages 371–388.

[5] Dominique Larchey-Wendling and Didier Galmiche. Nondeterministic phase semantics and the undecidability of boolean bi. *ACM Trans. Comput. Logic*, 14(1):6:1–6:41, February 2013.

[6] Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. In *FOCS*, volume 2, pages 662–671. IEEE, 1990.

[7] Michael Norrish. Mechanised Computability Theory. In *Proceedings of the Second International Conference on Interactive Theorem Proving*, ITP'11, pages 297–311, Berlin, Heidelberg, 2011. Springer-Verlag.

[8] Jian Xu, Xingyuan Zhang, and Christian Urban. Mechanising Turing Machines and Computability Theory in Isabelle/HOL. In *Interactive Theorem Proving, ITP 2013*, volume 7998 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 2013.