Proof Verification Can Be Hard!

Naveen Sundar Govindarajulu[2] & Selmer Bringsjord[1,2]

Department of Computer Science[1],Department of Cognitive Science[2]
Rensselaer Polytechnic Institute,
Troy NY 12180 USA
govinn2@rpi.edu • selmer@rpi.edu

Keywords: restricted ω-rule, not-semi-decidable, limits of proof verification

The generally accepted wisdom in computational circles is that pure proof verification is a solved problem and that the computationally hard elements and fertile areas of study lie in the domain of proof discovery. This holds for conventional proof systems such as first-order logic with a standard proof calculus such as natural deduction or resolution. This folk belief breaks down when we consider more user-friendly/powerful inference rules. One such rule is the **restricted ω-rule**, which is *not even semi-decidable* when added to a standard proof calculus of a *nice* theory.[1] While this might not be a novel result, we feel that the hardness of proof verification is under appreciated in most communities dealing with proofs. A proof sketch follows.

We set some context before we delve into the proof. The formal machinery and conventions follow (Ebbinghaus et al. 1984). We assume that we have the standard apparatus of first-order logic and that we are concerned only with theories of arithmetic, and machine checking and discovery of proofs of theorems of arithmetic. A theory Γ is said to be *negation-incomplete (incomplete)* iff there is at least one ϕ such that $\Gamma \nvdash \phi$ and $\Gamma \nvdash \neg\phi$. As most readers will recall, Gödel's first incompleteness theorem states that any sufficiently strong theory of arithmetic that has certain desired attributes is incomplete. Peano Arithmetic (PA) is one of the smallest incomplete theories that covers all of standard arithmetic. One way to address this shortcoming of incompleteness is to add more user-friendly (or mathematician-friendly) rules of inference.

The ω-rule is one such rule of inference to be used with arithmetic theories. The ω-rule is to be added to complete proof calculi. The ω-rule renders PA complete. This infinitary rule is of the following form:

$$\frac{\phi(\overline{0}), \phi(\overline{1}) \ldots,}{\forall x\, \phi(x)}\ \omega\text{-rule}$$

The above rule has an infinite number of premises and is clearly not suitable for implementation. A *restricted* ω-*rule* is a finite form of the rule which still keeps PA complete.

Assume that we have machines operating over representations of numerals and proofs. Then if we have a machine m_ϕ, which for all $n \in \mathbb{N}$ and the formula ϕ with one free variable, produces a proof of $\phi(\overline{n})$ from some set of axioms Γ. That is, $\mathsf{m}_\phi : \overline{n} \mapsto \rho(\Gamma, \phi(\overline{n}))$. [2]

Given this, one form of the restricted ω-rule is as follows:

$$\frac{\Gamma \quad \mathsf{m}_\phi}{\forall x\, \phi(x)}$$

Though the restricted ω-rule can be written down, full checking of the rule is beyond any machine implementation, as in the general case, a proof verification system that handles the rule should be able to check in all possible cases if the program supplied halts with the correct proof. A simple proof of this limit is given in the appendix. We feel that this limitative result demonstrates that proof representation and proof verification in mathematics can be a fertile area of study involving a rich interplay between expressibility and computational costs.

[1] *Nice* theories are consistent, decidable, and allow representations (Ebbinghaus, Flum & Thomas 1984). Roughly put, if a theory allows representations, it can prove facts about the primitive-recursive relations and functions. (See (Smith 2007).) A formal system (a theory Γ and a proof calculus ρ) is decidable/semi-decidable/not-semi-decidable if the decision problem $\Gamma \vdash_\rho \gamma$ is decidable/semi-decidable/not-semi-decidable.

[2] The most accessible reference for the ω-rule is (Baker, Ireland & Smaill 1992). All these results, except the main theorem in this abstract, are available in (Ebbinghaus et al. 1984, Franzén 2004).

Appendix: Proof

Theorem: $\langle \mathsf{PA}, \rho^{\omega} \rangle$ is not-semi-decidable

Proof. Let $\langle \mathsf{PA}, \rho^{\omega} \rangle$ denote the formal system comprised of PA with a standard proof calculus ρ augmented with the restricted ω-rule. Assume that we are only talking about Turing machines which output exactly one of $\{\mathsf{yes}, \mathsf{no}\}$ on all inputs or go on forever without halting, i.e., loops. The inputs are numerals which encode natural numbers.

Given: $\langle \mathsf{PA}, \rho^{\omega} \rangle$ is negation-complete and all its theorems are true on the standard model on $\langle \mathbb{N}; 0, \mathsf{S}, +, 1 \rangle$.

The following three statements can be coded up as arithmetic statements in the language of PA.

1. Machine m on input n halts with yes
2. Machine m on input n halts with no
3. Machine m on input n loops

For any machine m and any input n exactly one of the above is true in the standard model and therefore a theorem in $\langle \mathsf{PA}, \rho^{\omega} \rangle$.

Assumption 1:$\langle \mathsf{PA}, \rho^{\omega} \rangle$ *is semi-decidable.* That is, we have a machine G which on input $\langle p, q \rangle$ outputs yes if p represents a proof in $\langle \mathsf{PA}, \rho^{\omega} \rangle$ of the statement encoded by q, otherwise outputs no or loops.

If **Assumption 1** holds, then we can have a machine H which on input $\langle m, n \rangle$ decides if machine m halts on input n, i.e., H solves the halting problem. The machine H is specified below as an algoritm.

Algorithm for Machine H

Input : $\langle m, n \rangle$
Output: Does m halt on n?
initialization;
init
  q_1= "Arithmetic Statement encoding that m on input n halts with yes";
  q_2= "Arithmetic Statement encoding that m on input n halts with no";
  q_3= "Arithmetic Statement encoding that m on input n does not halt or loops.";
H is composed of three parallel threads, exactly one of which halts. If any of the threads halts, H halts.
Thread 1
  Do a breadth-first search for a proof p such that G on $\langle p, q_1 \rangle$ halts with yes
Thread 2
  Do a breadth-first search for a proof p such that G on $\langle p, q_2 \rangle$ halts with yes
Thread 3
  Do a breadth-first search for a proof p such that G on $\langle p, q_3 \rangle$ halts with yes

Algorithm 1: Program H

Algorithm for Breadth-First Search for Finding a Proof for q

Assume we have an lexicographic ordering of strings $\langle p_0, p_1, ... \rangle$. In the first iteration we run G on $\langle p_0, q \rangle$ for one step. In the next iteration, we run G $\langle p_0, q \rangle$ for one more step and we also run G on $\langle p_1, q \rangle$ for one step. We keep continuing in this fashion till we hit a p such that G on $\langle p, q \rangle$ stops with yes.

Algorithm 2: Breadth-First Search for a Proof

One of the three threads in H will halt. Therefore H decides the halting problem. We have arrived at a contradiction by assuming **Assumption 1**, which can be now be discarded, proving the our main thesis. ■

References

Baker, S., Ireland, A. & Smaill, A. (1992), On the Use of the Constructive Omega-rule within Automated Deduction, *in* 'Logic Programming and Automated Reasoning', Springer, pp. 214–225.

Ebbinghaus, H. D., Flum, J. & Thomas, W. (1984), *Mathematical Logic*, Springer-Verlag, New York, NY.

Franzén, T. (2004), 'Transfinite Progressions: A Second Look at Completeness', *Bulletin of Symbolic Logic* pp. 367–389.

Smith, P. (2007), *An Introduction to Gödel's Theorems*, Cambridge University Press, Cambridge, UK.