arXiv:0901.2551v2 [math.LO] 12 Mar 2009

The computational content of classical arithmetic*

Jeremy Avigad

March 12, 2009

Dedicated to Grigori Mints in honor of his seventieth birthday.

Abstract

Almost from the inception of Hilbert's program, foundational and structural efforts in proof theory have been directed towards the goal of clarifying the computational content of modern mathematical methods. This essay surveys various methods of extracting computational information from proofs in classical first-order arithmetic, and reflects on some of the relationships between them. Variants of the Gödel-Gentzen double-negation translation, some not so well known, serve to provide canonical and efficient computational interpretations of that theory.

1 Introduction

Hilbert's program was launched, in 1922, with the specific goal of demonstrating the consistency of modern, set-theoretic methods, using only finitary means. But the program can be viewed more broadly as a response to the radical methodological changes that had been introduced to mathematics in the late nineteenth century. Central to these changes was a shift in mathematical thought whereby the goal of mathematics was no longer viewed as that of developing powerful methods of calculation, but, rather, that of characterizing abstract, possibly infinite, mathematical structures, often in ways that could not easily be reconciled with a computational understanding.

Grisha's work over the years has touched on almost every aspect of proof theory, both of the reductive (foundational) and structural sort, involving a wide range of logical frameworks. But much of his work addresses the core proof-theoretic concern just raised, and has served to provide us with a deep and satisfying understanding of the computational content of nonconstructive, infinitary reasoning. Such work includes his characterization of the provably total computable functions of $I\Sigma_1$ as exactly the primitive recursive functions

*This work has been partially supported by NSF grant DMS-0700174 and a grant from the John Templeton Foundation.

([28, 29]); his method of continuous cut elimination, which provides a finitary interpretation of infinitary cut-elimination methods ([32, 11]); and his work on the epsilon substitution method (for example, [35, 37]).

Grisha has also been a friend and mentor to me throughout my career. The characterization of the provably total computable functions of $I\Sigma_1$ just mentioned was, in fact, also discovered by Charles Parsons and Gaisi Takeuti, all independently. I shudder to recall that at a meeting at Oberwolfach in 1998, when I was just two-and-a-half years out of graduate school, I referred to the result as "Parsons' theorem" in a talk before an audience that, unfortunately, included only the other two. Grisha asked the first question after the talk was over, and nothing in his manner or tone even hinted that I had made a *faux pas* (it didn't even occur to me until much later). In fact, I still vividly remember his encouraging and insightful comments, then and in later discussion. (For the record, Gaisi was equally gracious and supportive.)

In this essay, I will discuss methods of interpreting classical first-order arithmetic, often called *Peano arithmetic* (*PA*), in computational terms. Although the study of *PA* was central to Hilbert's program, it may initially seem like a toy theory, or an artificially simple case study. After all, mathematics deals with much more than the natural numbers, and there is a lot more to mathematical argumentation than the principle of induction. But experience has shown that the simplicity of the theory is deceptive: via direct interpretation or more elaborate forms of proof-theoretic reduction, vast portions of mathematical reasoning can be understood in terms of *PA* [4, 13, 43].

Here, I will be concerned with the Π_2, or "computational," consequences of *PA*. Suppose *PA* proves $\forall x\ \exists y\ R(x, y)$, where x and y range over the natural numbers and $R(x, y)$ is a decidable (say, primitive recursive) relation. We would like to understand how and to what extent such a proof provides an *algorithm* for producing such a y from a given x, one that is more informative than blind search. There are four methods that are commonly used to extract such an algorithm:

1. Gödel's *Dialectica* interpretation [18, 6], in conjunction with a double-negation interpretation that interprets *PA* in its intuitionistic counterpart, Heyting arithmetic (*HA*)

2. realizability [22, 23, 27, 45], again in conjunction with a double-negation translation, and either the Friedman A-translation [14] (often also attributed to Dragalin and Leivant, independently) or a method due to Coquand and Hofmann ([12, 1]) to "repair" translated Π_2 assertions

3. cut elimination ([15]; see, for example, [41])

4. the epsilon substitution method ([19, 8])

These four approaches really come in two pairs: the Dialectica interpretation and realizability have much in common, and, indeed, Paulo Oliva [39] has recently shown that one can interpolate a range of methods between the two; and,

similarly, cut elimination and the epsilon substitution method have a lot in common, as work by Grisha (e.g. [36]) shows. That is not to say that there aren't significant differences between the methods in each pairing, but the differences between the two pairs are much more pronounced.

For one thing, they produce two distinct sorts of "algorithms." The Dialectica interpretation, and Kreisel's "modified" version of Kleene's realizability, extract terms in Gödel's calculus of primitive recursive functionals of finite type, denoted PR^ω in Section 2 below. In contrast, cut elimination and the epsilon substitution method provide iterative procedures, whose termination can be proved by ordinal analysis. Specifically, one assigns (a notation for) an ordinal less than ε_0 to each stage of the computation in such a way that the ordinals decrease as the computation proceeds. Terms in PR^ω and $\prec\varepsilon_0$-recursive algorithms both have computational meaning, and there are various ways to "see" that the computations terminate; but, of course, any means of proving termination formally for all such terms and algorithms has to go beyond the means of *PA*.

Second, as indicated above, the first two methods involve an intermediate translation to *HA*, while the second two do not. It is true that the Dialectica interpretation and realizability can be applied to classical calculi directly (see [42] for the Dialectica interpretation, and, for example, [2, 38] for realizability); but I know of no such interpretation that cannot be understood in terms of a passage through intuitionistic arithmetic [2, 5, 44]. In contrast, cut elimination and the epsilon substitution method apply to classical logic directly. That is not to deny that one can apply cut elimination methods to intuitionistic logic (see, for example, [46]); but the arguments tend to be easier and more natural in the classical setting.

Finally, there is the issue of canonicity. Algorithms extracted from proofs in intuitionistic arithmetic tend to produce canonical witnesses to Π_2 assertions; work by Grisha [33, 34] shows, for example, that algorithms extracted by various methods yield the same results. In contrast, different ways of extracting witnesses from classical proofs yield different results, conveying the impression that there is something "nondeterministic" about classical logic. (There is a very nice discussion of this in [47, 48]. See also the discussion in Section 6 below.) Insofar as one has a natural translation from classical arithmetic to intuitionistic arithmetic, some of the canonicity of the associated computation is transferred to the former theory.

In this essay, I will discuss realizability and the Dialectica interpretation, as they apply to classical arithmetic, via translations to intuitionistic arithmetic. After reviewing some preliminaries in Section 2, I will discuss variations of the double-negation interpretation in Section 3. One, in particular, is very efficient when it comes to introducing negations; in Section 4, I will show that, when combined with realizability or the Dialectica interpretation, this yields computational interpretations of classical arithmetic that are efficient in their use of higher types. In Section 5, I will consider another curious double-negation interpretation, and diagnose an unfortunate aspect of its behavior.

I am grateful to Philipp Gerhardy, Thomas Streicher, and an anonymous

referee for helpful comments and corrections.

2 Preliminaries

Somewhat imprecisely, one can think of intuitionistic logic as classical logic without the law of the excluded middle; and one can think of minimal logic as intuitionistic logic without the rule *ex falso sequitur quodlibet*, that is, from $\bot$ conclude anything. Computational interpretations of classical logic often pass through minimal logic, which has the nicest computational interpretation. (One can interpret intuitionistic logic in minimal logic by replacing every atomic formula A by $A \vee \bot$, so the difference between these two is small.)

To have a uniform basic to compare the different logics, it is useful to take the first-order logical symbols to be $\forall$, $\exists$, $\wedge$, $\vee$, $\rightarrow$, and $\bot$, with $\neg\varphi$ defined to be $\varphi \rightarrow \bot$. However, when it comes to classical logic, it is often natural to restrict one's attention to formulas in *negation-normal form*, where formulas are built up from atomic and negated atomic formulas using $\wedge$, $\vee$, $\forall$, and $\exists$. A negation operator, $\sim\varphi$, can be defined for such formulas; $\sim\varphi$ is what you get if, in φ, you exchange $\wedge$ with $\vee$, $\forall$ with $\exists$, and atomic formulas with their negations. Note that $\sim\sim\varphi$ is just φ. Classically, every formula φ has a negation-normal form equivalent, φ^{nnf}, obtained by defining $(\theta \rightarrow \eta)^{\mathrm{nnf}}$ to be $\sim\theta^{\mathrm{nnf}} \vee \eta^{\mathrm{nnf}}$, and treating the other connectives in the obvious way. This has the slightly awkward consequence that $(\neg\varphi)^{\mathrm{nnf}}$ translates to $\sim\varphi^{\mathrm{nnf}} \vee \bot$, but simplifying $\theta \vee \bot$ to θ and $\theta \wedge \bot$ to $\bot$ easily remedies this.

There are a number of reasons why negation-normal form is so natural for classical logic. First of all, it is easy to keep track of polarities: if φ is in negation-normal form, then every subformula is a positive subformula, except for, perhaps, atomic formulas; an atomic formula A occurs positively in φ if it occurs un-negated, and negatively if it occurs with a negation sign before it. Second, the representation accords well with practice: any classically-minded mathematician would not hesitate to prove "if φ then ψ" by assuming $\neg\psi$ and deriving $\neg\varphi$, or by assuming φ and $\neg\psi$ and deriving a contradiction; so it is convenient that $\varphi \rightarrow \psi$, $\neg\psi \rightarrow \neg\varphi$, and $\neg(\varphi \wedge \neg\psi)$ have the same negation-normal form representation. Finally, proof systems for formulas in negation-normal form tend to be remarkably simple (see, for example, [41, 46]).

It was Gödel [18] who first showed that the provably total computable functions of arithmetic can be characterized in terms of the primitive recursive functionals of finite type (see [6, 20]). The set of finite types can be defined to be the smallest set containing the symbol N, and closed under an operation which takes types σ and τ to a new type $\sigma \rightarrow \tau$. In the intended ("full") interpretation, N denotes the set of natural numbers, and $\sigma \rightarrow \tau$ denotes the set of all functions from σ to τ. A set of terms, PR^ω, is defined inductively as follows:

1. For each type σ, there is a stock of variables $x, y, z, \ldots$ of type σ.
2. 0 is a term of type N.

3. S (successor) is a term of type $\mathsf{N} \to \mathsf{N}$.

4. if s is a term of type $\tau \to \sigma$ and t is a term of type τ, then $s(t)$ is a term of type σ.

5. if s is a term of type σ and x is a variable of type τ, then $\lambda x\ s$ is a term of type $\tau \to \sigma$.

6. If s is a term of type σ, and t is a term of type $\mathsf{N} \to (\sigma \to \sigma)$, then R_{st} is a term of type $\mathsf{N} \to \sigma$.

Intuitively, $s(t)$ denotes the result of applying s to t, $\lambda x\ s$ denotes the function which takes any value of x to s, and R_{st} denotes the function defined from s and t by primitive recursion, with $R_{st}(0) = s$ and $R_{st}(S(x)) = t(x, R_{st}(x))$ for every x. In this last equation, I have adopted the convention of writing $t(r, s)$ instead of $(t(r))(s)$.

It will be convenient below to augment the finite types with products $\sigma \times \tau$, associated pairing operations $(\cdot, \cdot)$, and projections $(\cdot)_0$ and $(\cdot)_1$. Product types can be eliminated in the usual way by currying and replacing terms t with sequences of terms t_i. It will also be convenient to have disjoint union types $\sigma + \tau$, an element of which is either an element of σ or an element of τ, tagged to indicate which is the case. That is, for each such type we have insertion operations, inl and inr, which convert elements of type σ and τ respectively to an element of type $\sigma + \tau$; predicates isleft(a) and isright(a), which indicate whether a is tagged to be of type σ or τ; and functions left(a) and right(a), which interpret a as an element of type σ and τ, respectively. References to such sum types can be eliminated by taking $\sigma + \tau$ to be $\mathsf{N} \times \sigma \times \tau$, defining $\mathrm{inl}(a) = (0, a, 0^\tau)$, defining $\mathrm{inr}(a) = (1, 0^\sigma, a)$, where 0^σ and 0^τ are constant zero functionals of type σ, τ respectively, and so on.

In the next section, I will describe various double-negation interpretations that serve to reduce classical arithmetic, *PA*, to intuitionistic arithmetic, *HA* — in fact, to *HA* taken over *minimal logic*. These show that if *PA* proves a Π_2 formula $\forall x\ \exists y\ R(x, y)$, then *HA* proves $\forall x\ \neg\neg\exists y\ R(x, y)$; in fact, a variant HA' of *HA* based on minimal logic suffices. This reduces the problem to extracting computational information from the latter proof.

One method of doing so involves using Kreisel's notion of modified realizability, combined with the Friedman A-translation. One can extend *HA* to a higher-type version, HA^ω, which has variables ranging over arbitrary types, and terms of all the primitive recursive functionals. Fix any primitive recursive relation $A(y)$; then to each formula $\varphi(\bar{x})$ in the language of arithmetic, one

inductively assigns a formula "a *realizes* $\varphi(\bar{x})$," as follows.

$$\begin{aligned}
a \textit{ realizes } \bot &\equiv A(a) \\
a \textit{ realizes } \theta &\equiv \theta, \text{ if } \theta \text{ is atomic} \\
a \textit{ realizes } \varphi \wedge \psi &\equiv ((a)_0 \textit{ realizes } \varphi) \wedge ((a)_1 \textit{ realizes } \varphi) \\
a \textit{ realizes } \varphi \vee \psi &\equiv ((\text{isleft}(a) \wedge \text{left}(a) \textit{ realizes } \varphi) \vee \\
&\qquad (\text{isright}(a) \wedge \text{right}(a) \textit{ realizes } \psi)) \\
a \textit{ realizes } \varphi \to \psi &\equiv \forall b\, (b \textit{ realizes } \varphi \to a(b) \textit{ realizes } \psi) \\
a \textit{ realizes } \forall x\, \varphi(x) &\equiv \forall x\, (a(x) \textit{ realizes } \varphi(x)) \\
a \textit{ realizes } \exists x\, \varphi(x) &\equiv (a)_1 \textit{ realizes } \varphi((a)_0)
\end{aligned}$$

Now, suppose classical arithmetic proves $\forall x\ \exists y\ R(x, y)$, for some primitive recursive relation R. Then, using a double-negation translation, HA' proves $\forall x\ \neg\neg\exists y\ R(x, y)$, and hence it proves $\neg\neg\exists y\ R(c, y)$ for a fresh constant c. Fix $A(y)$ in the realizability relation above to be the formula $R(c, y)$. Inductively, one can then extract from the proof of term t of PR^ω such that HA^ω proves that t realizes $\neg\neg\exists y\ R(x, y)$. Now notice that the identity function, id, realizes $\neg\exists y\ R(c, y)$, since a realizer to $\exists y\ R(c, y)$ is simply a value of a satisfying $R(c, a)$. Thus if a realizes $\exists y\ R(c, y)$, then $a(id)$ satisfies $R(c, a(id))$. Viewing a, now, as a function of c, yields the following conclusions:

Theorem 2.1. *If classical arithmetic proves $\forall x\ \neg\neg\exists y\ R(x, y)$, there is a term F of PR^ω of type N to N such that HA^ω proves $\forall x\ R(x, F(x))$.*

See [45, 24] for more about realizability, and [14] for the A-translation.

Gödel's Dialectica interpretation provides an alternative route to this result. In fact, one obtains a stronger conclusion, namely that the correctness of the witnessing term can be proved in a quantifier-free fragment PR^ω of HA^ω. To each formula φ in the language of arithmetic, one inductively assigns a formula φ^D of the form $\exists x\ \forall y\ \varphi_D(x, y)$, where x and y are now tuples of variables of appropriate types. Assuming $\psi^D = \exists u\ \forall v\ \psi_D(u, v)$, the assignment is defined as follows:

$$\begin{aligned}
\theta^D &\equiv \theta, \text{ if } \theta \text{ is atomic} \\
(\varphi \wedge \psi)^D &\equiv \exists x, u\ \forall y, v\ (\varphi_D \wedge \psi_D) \\
(\varphi \vee \psi)^D &\equiv \exists z\ \forall y, v\ (\text{isleft}(z) \wedge \varphi_D(\text{left}(z), y) \vee \\
&\qquad (\text{isright}(z) \wedge \psi_D(\text{right}(z), v))) \\
(\varphi \to \psi)^D &\equiv \exists U, Y\ \forall x, v\ (\varphi_D(x, Y(x, v)) \to \psi_D(U(x), v)) \\
(\forall z\ \varphi(z))^D &\equiv \exists X\ \forall z, y\ \varphi_D(X(z), y, z) \\
(\exists z\ \varphi(z))^D &\equiv \exists z, x\ \forall y\ \varphi_D(x, y, z)
\end{aligned}$$

The clause for implication is the most interesting among these, and can be understood as follows: from a witness, x, to the hypothesis, $U(x)$ is supposed to return a witness to the conclusion; and given a purported counterexample,

v, to the conclusion, $Y(x, v)$ is supposed to return a counterexample to the hypothesis. Since we have defined $\neg\varphi$ to be $\varphi \to \bot$, notice that $(\neg\varphi)^D$ is $\exists Y\ \forall x\ \neg\varphi_D(x, Y(x))$.

The Dialectica interpretation of $\forall x\ \neg\neg\exists y\ R(x, y)$ is $\exists Y\ \forall x\ \neg\neg R(x, Y(x))$, which is intuitionistically equivalent to $\exists Y\ \forall x\ R(x, Y(x))$, given the decidability of primitive recursive relations. One can show that from a proof of φ in *HA*, one can extract a term F such that for every x, PR^ω proves $\varphi_D(x, F(x))$, once again yielding Theorem 2.1.

3 Some double-negation translations

We have seen that one can use modified realizability or the Dialectica interpretation to extract algorithm from a proof of a Π_2 statement in classical arithmetic, modulo a method of reducing classical arithmetic to intuitionistic arithmetic. Double negation translations provide the latter.

A formula is said to be *negative* if it does not involve $\exists$ or $\vee$ and each atomic formula A occurs in the form $\neg A$; in other words, the formula is built up from negated atomic formula using $\forall$, $\wedge$, $\to$, and $\bot$. Over minimal logic, negative formulas are stable under double negation, which is to say, if φ is any negative formula, then *HA* proves that $\neg\neg\varphi$ is equivalent to φ (see, for example, [46]).

The Gödel-Gentzen double-negation translation maps an arbitrary first-order formula φ to a negative formula, φ^N:

$$\begin{aligned}
\bot^N &\equiv \bot \\
\theta^N &\equiv \neg\neg\theta, \text{ if } \theta \text{ is atomic} \\
(\varphi \wedge \psi)^N &\equiv \varphi^N \wedge \psi^N \\
(\varphi \vee \psi)^N &\equiv \neg(\neg\varphi^N \wedge \neg\psi^N) \\
(\varphi \to \psi)^N &\equiv \varphi^N \to \psi^N \\
(\forall x\ \varphi)^N &\equiv \forall x\ \varphi^N \\
(\exists x\ \varphi)^N &\equiv \neg\forall x\ \neg\varphi^N
\end{aligned}$$

The translation has the following properties:

Theorem 3.1. *For any formula φ and set of sentences Γ:*

1. *Classical logic proves $\varphi \leftrightarrow \varphi^N$*
2. *If φ is provable from Γ in classical logic, then φ^N is provable from Γ^N in minimal logic.*

Since the *HA* proves the $\cdot^N$ translations of its own axioms, we have as a corollary:

Corollary 3.2. *If PA proves φ, then HA proves φ^N.*

In fact, one can strengthen the corollary in three ways:

1. Since HA proves $\neg\neg\theta \to \theta$ for atomic formulas θ, one can define θ^N to be θ.

2. Assuming the language of HA includes, say, symbols denoting the primitive recursive functions, every negated atomic formula, $\neg\theta$, has an atomic equivalent, $\bar{\theta}$; so one can define $(\neg\theta)^N$ to be $\bar{\theta}$.

3. The theorem remains true if one replaces HA by a suitable variant, HA', based on minimal logic.

These considerations hold in the theorems that follow as well.

The reason to be concerned about negations is that they are undesirable with respect to the two computational interpretations given in the last section, since they lead to the use of more complicated types in the resulting terms of PR^ω. There is a variant of the double-negation translation known as the *Kuroda translation* that fares slightly better in this regard: for any formula φ, let φ^{Ku} denote the result of doubly-negating atomic formulas, and adding a double negation *after* each universal quantifier, and, finally, adding a double-negation to the front of the formula. Then we have:

Theorem 3.3. *For every formula φ, $\varphi^{Ku} \leftrightarrow \varphi^G$ is provable in minimal logic. Hence PA proves φ if and only if HA proves φ^{Ku}.*

Note that intuitionistic logic, rather than minimal logic, is required in the conclusion.

Late in 2005, Grisha asked whether a version of the Dialectica interpretation designed by Shoenfield [42], for classical arithmetic, could be understood as a composition of the usual Dialectica interpretation together with a double-negation translation. I set the question aside and solved it a few months later [5], only to find that Ulrich Kohlenbach and Thomas Streicher had solved it more quickly [44]. In a way that can be made precise, the Shoenfield translation corresponds to the following version of the double-negation interpretation (itself a variant of a translation due to Krivine), expressed for a basis involving the connectives $\neg$, $\wedge$, $\vee$, and $\forall$. We define φ^{Kr} to be $\neg\varphi_{Kr}$, where φ_{Kr} is defined recursively by clauses below. It helps to keep in mind that φ_{Kr} is supposed to represent the *negation* of φ:

$$\begin{aligned}
\theta_{Kr} &\equiv \neg\theta, \text{ if } \theta \text{ is atomic} \\
(\neg\varphi)_{Kr} &\equiv \neg\varphi_{Kr} \\
(\varphi \wedge \psi)_{Kr} &\equiv \varphi_{Kr} \vee \psi_{Kr} \\
(\varphi \vee \psi)_{Kr} &\equiv \varphi_{Kr} \wedge \psi_{Kr} \\
(\forall x\, \varphi)_{Kr} &\equiv \exists x\, \varphi_{Kr}
\end{aligned}$$

Note that we can eliminate either $\vee$ or $\wedge$ and retain a complete set of connectives, but including them both is more efficient. Formulas of the form $\exists x\, \varphi$, however, have to be expressed as $\neg\forall x\, \neg\varphi$ to apply the translation.

Theorem 3.4. *For every formula φ, $\varphi^{Kr} \leftrightarrow \varphi^G$ is provable in minimal logic. Hence PA proves φ if and only if HA′ proves φ^{Kr}.*

The $\cdot^{Kr}$-translation is particularly good when it comes to formulas in negation-normal form; it only adds two quantifiers for each existential quantifier, as well as one at the beginning. But one can do even better [2]. Taking advantage of the classical negation operator, now φ^M is defined to be $\neg(\sim\varphi)_M$, where the map $\psi \mapsto \psi_M$ is defined recursively as follows:

$$\begin{aligned}
\theta_M &\equiv \theta, \text{ if } \theta \text{ is atomic or negated atomic} \\
(\varphi \vee \psi)_M &\equiv \varphi_M \vee \psi_M \\
(\varphi \wedge \psi)_M &\equiv \varphi_M \wedge \psi_M \\
(\exists x\, \varphi)_M &\equiv \exists x\, \varphi_M \\
(\forall x\, \varphi)_M &\equiv \neg\exists x\, (\sim\varphi)_M.
\end{aligned}$$

Once again, we have

Theorem 3.5. *For every formula φ in negation-normal form, $\varphi^M \leftrightarrow \varphi^G$ is provable in minimal logic. Hence PA proves φ if and only if HA′ proves φ^M.*

The $\cdot^M$-translation is extremely efficient with respect to negations, introducing, roughly, one at the beginning of the formula, and one for every quantifier alternation after an initial block of universal quantifiers. Alternatively, can define $(\varphi \wedge \psi)_M$ in analogy to $(\forall x\, \varphi)_M$, as $\neg((\sim\varphi)_M \vee (\sim\psi)_M)$. This gives the translation the nice property that given the formulas φ^M and $(\sim\varphi)^M$, one is the negation of the other. But there is a lot to be said for keeping negations to a minimum.

Of course, the $\cdot^M$-translation extends to all classical formulas by identifying them with their canonical negation-normal form equivalents. Since the translation relies on the negation-normal form representation of classical formulas, it shares many nice properties with a more complicated double-negation translation due to Girard [17]. It is this translation that I will use, in the next section, to provide an efficient computational interpretation of classical arithmetic.

Let me close with one more translation, found in [3], which is interesting in its own right. For reasons that will become clear later on, I will call it "the awkward translation": if φ is any formula in negation-normal form, let φ^{awk} denote $\neg(\sim\varphi)$.

Theorem 3.6. *For any formula φ, $\varphi^G \to \varphi^{awk}$ is provable in minimal logic. Hence, PA proves φ if and only if HA′ proves φ^{awk}.*

Proof. Once can show by induction that if ψ is any formula in negation-normal form, then $\psi \to \psi^G$ is provable in minimal logic. So minimal logic proves that $\sim\varphi$ implies $(\sim\varphi)^G$, and hence that $\neg(\sim\varphi)^G$ implies φ^{awk}. But since $\sim\varphi$ is classically equivalent to $\neg\varphi$, $\neg(\sim\varphi)^G$ is equivalent to $\neg\neg\varphi^G$, which is implied by φ^G. □

The $\cdot^{awk}$-translation is almost absurdly efficient with respect to negations: the one classical negation on the inside adds no negations at all (recall that in arithmetic, negated atomic formulas have atomic equivalents), and the translation adds only one negation on the outside. But the attentive reader will have noticed that the first assertion in Theorem 3.6 is slightly weaker than the corresponding assertions in the the theorems that precede it: only one direction of the equivalence is minimally valid. We will see, in Section 5, that this means that the translation fares very poorly with respect to modus ponens, making it impossible to translate ordinary proofs piece by piece.

For pure first-order logic, an alternative proof of Theorem 3.6 can be found in [3]. Benno van den Berg (personal communication) later hit upon this same translation, independently. In [3], I claimed that with intuitionistic logic in place of minimal logic, the result is a consequence of a characterization of Glivenko formulas due to Orevkov, described in a very nice survey [30, Section 3.2.5] of Russian proof theory by Grisha.[1] That seems to be incorrect; but van den Berg and Streicher have pointed out to me that in that case the result follows a theorem due to Mints and Orevkov [30, page 404, paragraph 4].

4 Interpreting classical arithmetic

We now obtain direct computational interpretations of classical arithmetic simply by combining the $\cdot^M$-translation of Section 3 with the computational interpretations of HA' given in Section 2. One annoying consequence of the use of the classical negation operator in the $\cdot^M$-translation is that it is impossible to carry out the translation of a formula φ from the inside out: depending on the context in which a subformula ψ occurs, the computational interpretation of the full formula may depend on either the computational interpretation of ψ or the computational interpretation of $\sim\psi$. In practice, then, it is often more convenient to carry out the interpretation in two steps, applying the $\cdot^M$-translation first, and then one of the two computational interpretations described in Section 2. Nonetheless, it is interesting to see what happens when the steps are composed, which is what I will do here. Both translations apply to formulas in negation-normal form, and we can assume that negated atomic subformulas are replaced by their atomic equivalents.

As in Section 2, the appropriate version of classical realizability is defined relative to a fixed primitive recursive predicate $A(y)$. Most of the clauses look

[1] There are typographical errors on page 401 of that paper, which Grisha has corrected for me. The last class eight lines from the bottom of the page should be $\{\rightarrow^-, \sim^-, \vee+, \exists^+\}$; the last class seven lines from the bottom of the page should be $\{\rightarrow^-, \sim^-, \vee+, \rightarrow^+, \forall^+\}$; and the first class at the bottom line should be $\{\rightarrow^+, \sim^+, \vee^-\}$.

just like ordinary modified realizability:

$$\begin{aligned}
a \ \textit{realizes} \ \theta &\equiv \theta, \text{ if } \theta \text{ is atomic} \\
a \ \textit{realizes} \ \varphi \wedge \psi &\equiv ((a)_0 \ \textit{realizes} \ \varphi) \wedge ((a)_1 \ \textit{realizes} \ \varphi) \\
a \ \textit{realizes} \ \varphi \vee \psi &\equiv ((\mathrm{isleft}(a) \wedge \mathrm{left}(a) \ \textit{realizes} \ \varphi) \vee \\
&\qquad (\mathrm{isright}(a) \wedge \mathrm{right}(a) \ \textit{realizes} \ \psi)) \\
a \ \textit{realizes} \ \exists x \, \varphi(x) &\equiv (a)_1 \ \textit{realizes} \ \varphi((a)_0)
\end{aligned}$$

The only slightly more complicated clause is the one for the universal quantifier. Take $a \ \textit{refutes} \ \varphi$ to be the formula $\forall b \, (b \ \textit{realizes} \ \varphi \to A(a(b)))$.

$$a \ \textit{realizes} \ \forall x \, \varphi(x) \equiv a \ \textit{refutes} \ \exists x \sim\varphi(x)$$

One can then straightforwardly extract, from any proof of a formula φ in classical arithmetic, a term a that refutes $\sim\varphi$. But now notice that the identity function realizes $\forall x \ \bar{A}(x)$, where $\bar{A}$ is the negation of A; so, from a proof of $\exists y \ A(y)$ in classical arithmetic, since the identity function realizes $\forall x \ \bar{A}(x)$, one obtains a term a satisfying $A(a)$. This provides a direct proof of Theorem 2.1. Details can be found in [2]. A more elaborate realizability relation, based on the A-translation, can be found in [9].

The corresponding variant of the Dialectica translation is similarly straightforward. As with the Shoenfield variant [42, 5], each formula φ is mapped to a formula $\varphi^{D'}$ of the form $\forall x \ \exists y \ \varphi_{D'}(x, y)$, where x and y are sequences of variables. Assuming $\psi^{D'}$ is $\forall u \ \exists v \ \psi_{D'}(u, v)$, the translation is defined recursively, as follows:

$$\begin{aligned}
\theta^{D'} &\equiv \theta, \text{ if } \theta \text{ is atomic} \\
(\varphi \wedge \psi)^{D'} &\equiv \forall x, u \ \exists y, v \ (\varphi_{D'}(x, y) \wedge \psi_{D'}(u, v)) \\
(\varphi \vee \psi)^{D'} &\equiv \forall x, u \ \exists y, v \ (\varphi_{D'}(x, y) \vee \psi_{D'}(u, v)) \\
(\forall z \ \varphi)^{D'} &\equiv \forall z, x \ \exists y \ \varphi_{D'}(x, y)
\end{aligned}$$

This time, it is the clause for the existential quantifier that is slightly more complicated. If $(\sim\varphi(z))^{D'}$ is $\forall r \ \exists s \ (\sim\varphi)_{D'}(z, r, s)$, define

$$(\exists z \ \varphi)^{D'} \equiv \forall S \ \exists z, r \ \neg(\sim\varphi)_{D'}(z, r, S(z, r)).$$

On can interpret this as saying that for any function $S(z, r)$ that purports to witness $\forall z, r \ \exists s \ (\sim\varphi)_{D'}(z, r, s)$, there are a z and an r denying that claim. Once again, one can straightforwardly extract, from any proof of a formula φ in classical arithmetic, a term a satisfying $\forall x \ \varphi_{D'}(x, a(x))$. This provides another direct proof of Theorem 2.1.

5 Back to the awkward translation

I would now like to come back to the "awkward translation," discussed at the end of Section 3. I will do this via what at first might seem to be a digression through

Kreisel's *no-counterexample interpretation*. Let φ be a formula in prenex form, for example,

$$\exists x\, \forall y\, \exists z\, \forall w\, \theta(x, y, z, w).$$

The *Herbrand normal form* φ^H of φ is obtained by replacing the universally quantified variables of φ by function symbols that depend on the preceding existential variables, to obtain

$$\exists x, z\, \theta(x, f(x), z, g(x, z)).$$

It is not hard to check that, in classical logic, φ implies φ^H. Thus, by a slight variant of Theorem 2.1 (relativizing it to function symbols), if classical arithmetic proves φ, there will be terms $F_1(f, g)$ and $F_2(f, g)$ of PR^ω such that HA^ω proves

$$\forall f, g\, \theta(F_1(f, g), f(F_1(f, g)), F_2(f, g), g(F_1(f, g), F_2(f, g))).$$

Think of f and g as providing purported counterexamples to the truth of φ, so that F_1 and F_2 effectively foil such counterexamples. The no-counterexample interpretation is simply the generalization of this transformation to arbitrary prenex formulas.

One need not invoke Herbrand normal form to arrive at the previous conclusion. One can check that if φ is a prenex formula, the no-counterexample interpretation of φ is essentially just the Dialectica interpretation of φ^{awk}, so the result follows from Theorem 3.6 as well.

The no-counterexample interpretation can be viewed as a computational interpretation of arithmetic. But, in a remarkable article [25], Kohlenbach has shown that it is not a very *modular* computational interpretation, in the sense that it does not have nice behavior with respect to modus ponens. To make this claim precise, note that the set of (terms denoting) primitive recursive functionals, PR^ω, can be stratified into increasing subsets PR^ω_n, in such a way that any finite fragment of *HA* has a Dialectica interpretation (or modified realizability interpretation) using only terms in that set. Kohlenbach [25, Proposition 2.2] shows:

Theorem 5.1. *For every n there are sentences φ and ψ of arithmetic such that:*

1. *φ is prenex.*
2. *ψ is a Π_2 sentence, that is, of the form $\forall x\, \exists y\, R(x, y)$ for some primitive recursive relation R.*
3. *Primitive recursive arithmetic proves φ.*
4. *PA proves $\varphi \to \psi$.*
5. *φ and every prenexation of $\varphi \to \psi$ has a no-counterexample interpretation with functionals in PR^ω_0.*

But:

6. *There is no term F of PR_n^ω which satisfies the no-counterexample interpretation of ψ; that is, there is no term F such that $\forall x\, R(x, F(x))$ is true in the standard model of arithmetic.*

Theorem 5.1 shows that there is no straightforward way to combine witnesses to the no-counterexample interpretations of φ and $\varphi \to \psi$, respectively, to obtain a witness to the no-counterexample interpretation of ψ.

The problem with the awkward translation is that, similarly, it may behave poorly with respect to modus ponens. Consider pure first-order logic with a single predicate symbol, $A(x)$. Then there are formulas φ and ψ such that ψ^{awk} doesn't follow from φ^{awk} and $(\varphi \to \psi)^{awk}$ in minimal logic: just take φ to be the formula $\forall x\, A(x)$ and ψ to be $\bot$. In that case, $\varphi^{awk} \wedge (\varphi \to \psi)^{awk} \to \psi^{awk}$ is equivalent, over minimal logic, to the double-negation shift, $\forall x\, \neg\neg A(x) \to \neg\neg A(x)$, which is not even provable in intuitionistic logic.

When I showed the awkward translation to Grisha, he remarked right away that its behavior has something to do with Kohlenbach's result. At the time, I had no idea what he meant; but writing this paper finally prodded me to sort it out. Grisha was right: Theorem 5.1 can, in fact, be used to show that the awkward translation does not provide a modular translation of Peano arithmetic to Heyting arithmetic, in the following sense.

Theorem 5.2. *For any fragment T of HA, there are formulas φ and ψ such that the following hold:*

1. *PA proves φ and $\varphi \to \psi$, but*
2. *T together with φ^{awk} and $(\varphi \to \psi)^{awk}$ does not prove ψ^{awk}.*

This shows that modus ponens fails under the $\cdot^{awk}$-translation, in a strong way. Since *PA* proves φ and $\varphi \to \psi$, it also proves ψ, and so by Theorem 3.6, *HA* proves ψ^{awk}. But having the translation φ^{awk} and $(\varphi \to \psi)^{awk}$ may not help much in obtaining such a proof of ψ^{awk}; indeed, obtaining a proof of ψ^{awk} from φ^{awk} and $(\varphi \to \psi)^{awk}$ may be no easier than simply proving ψ^{awk} outright.

Proof. Given T, first let n be large enough so that the Dialectica interpretation of T uses only terms in PR_n^ω, and then let φ and ψ be as in Theorem 5.1. If there were a proof of ψ^{awk} from φ^{awk} and $(\varphi \to \psi)^{awk}$ in T, applying the Dialectica interpretation, one would obtain terms witnessing the Dialectica interpretation of ψ^{awk} from terms witnessing the Dialectica interpretations of φ^{awk} and $(\varphi \to \psi)^{awk}$. But the Dialectica interpretation of φ^{awk} is the no-counterexample interpretation of φ, and it is not hard to check that the Dialectica interpretation of $(\varphi \to \psi)^{awk}$ is the no-counterexample interpretation of one of the prenexations of $\varphi \to \psi$. Thus there would be a witness to the no-counterexample interpretation of ψ in PR_n^ω, contrary to the choice of φ and ψ. □

6 Conclusions

As noted in the introduction, conventional wisdom holds that classical logic is "nondeterministic," in that different ways of extracting algorithms from classical proofs can yield different results. Sometimes, however, nondeterminacy is unavoidable. For example, even minimal logic can prove $\exists x\ A(x) \wedge \exists y\ A(y) \to \exists z\ A(z)$, and any computational interpretation of this formula will have to choose either x or y to witness the conclusion. The difference is that this sentence is typically not taken to be an *axiom* of minimal logic; rather, there are two axioms, $\varphi \wedge \psi \to \varphi$ and $\varphi \wedge \psi \to \psi$, and any proof of the sentence has to choose one or the other. In contrast, standard calculi for classical logic provide cases where there are multiple choices of witnesses, with no principled reason to choose one over the other. For example, starting from canonical proofs of $A(a) \to \exists x\ A(x)$ and $A(b) \to \exists x\ A(x)$, one can weaken the conclusions to obtain proofs of $\varphi \to (A(a) \to \exists x\ A(x))$ and $\neg\varphi \to (A(b) \to \exists x\ A(x))$, for an arbitrary formula, φ. Then, using the law of the excluded middle, $\varphi \vee \neg\varphi$, one can combine these to obtain a proof of $A(a) \wedge A(b) \to \exists x\ A(x)$ where there is little reason to favor a or b as the implicit witness to the existential quantifier. (This example is essentially that given by Lafont [16, p. 150].)

This shows that standard classical calculi are nondeterministic in a way that proofs in intuitionistic and minimal logic are not. Girard [17] has neatly diagnosed the source of the nondeterminacy, and has provided a calculus for classical logic that eliminates it by forcing the prover to make an explicit choice in exactly those situations where an ambiguity would otherwise arise.[2] But note that the realizability interpretation of Section 4 also avoids this nondeterminacy; the translation procedure described in [2] is fully explicit and unambiguous. With respect to the example discussed in the last paragraph, the interpretation chooses a or b based on the logical form of φ.[3] Indeed, the results of [2] show that the witnesses obtained in this way coincide with those obtained using a natural class of cut-elimination procedures.

Yet another response to the example above is that of Urban and Bierman [47, 48], who simply embrace the nondeterminism as an inherent part of the computational interpretation of classical logic. In fact, Urban [47] has provided a nondeterministic programming language to interpret classical logic in a natural way. Passing through a double-negation interpretation, as we have done here, amounts to making specific choices to resolve the nondeterminism. It is an open-ended conceptual problem to understand which deterministic instances of the general nondeterministic algorithms can be realized in such a way.

[2] One way to understand what is going on is to notice that in minimal logic there are two distinct ways of proving $\neg\neg(\varphi \wedge \psi)$ from $\neg\neg\varphi$ and $\neg\neg\psi$, and this inference is needed to verify the classical axiom $\neg\neg\theta \to \theta$ under the double-negation translation. Another way is to notice that since, in classical logic, φ and $\neg\neg\varphi$ are equivalent, there is little reason to favor φ or $\neg\varphi$ in situations where intuitionistic logic treats them differently. I am grateful to Thomas Streicher for these insights.

[3] More generally, what breaks the symmetry alluded to in the previous footnote is that φ and $\neg\neg\varphi$ have the same negation-normal form, which is distinct from (and dual to) that of $\neg\varphi$.

At this point, however, we should be clearer as to the goals of our analysis. Ordinary mathematical proofs are not written in formal languages, and so the process of extracting an algorithm from even a rather constructive mathematical argument can involve nondeterminism of sorts. And, despite some interesting explorations in this direction [10], it is far from clear that classical arithmetic can be used as an effective programming language in its own right. But formal methods are actively being developed in support of software verification [21], and a better understanding of the computational content of classical logic may support the development of better logical frameworks for that purpose [40]. Formal translations like the ones described here have also been effective in "proof mining," the practice of using logical methods to extract mathematically useful information from nonconstructive proofs [7, 24, 26].

Grisha's work has, primarily, addressed the general foundational question as to the computational content of classical methods. In that respect, the general metatheorems described here provide a satisfying answer: for the most part, classical mathematical reasoning *does* have computational content, which is to say, algorithms can be extracted from classical proofs; but by suppressing computational detail, the proofs often leave algorithmic detail underspecified, rendering them amenable to different implementations. Grisha's work has thus contributed to an understanding of the computational content of classical arithmetic that is mathematically and philosophically satisfying, providing a solid basis for further scientific research.

References

[1] Jeremy Avigad. Interpreting classical theories in constructive ones. *J. Symbolic Logic*, 65:1785–1812, 2000.

[2] Jeremy Avigad. A realizability interpretation for classical arithmetic. In *Logic Colloquium '98 (Prague)*, pages 57–90. Assoc. Symbol. Logic, Urbana, IL, 2000.

[3] Jeremy Avigad. Algebraic proofs of cut elimination. *J. Log. Algebr. Program.*, 49:15–30, 2001.

[4] Jeremy Avigad. Number theory and elementary arithmetic. *Philosophia Mathematica*, 11:257–284, 2003.

[5] Jeremy Avigad. A variant of the double-negation translation. Carnegie Mellon Technical Report CMU-PHIL 179.

[6] Jeremy Avigad and Solomon Feferman. Gödel's functional ("Dialectica") interpretation. In *Handbook of proof theory*, pages 337–405. North-Holland, Amsterdam, 1998.

[7] Jeremy Avigad, Philipp Gerhardy, and Henry Towsner. Local stability of ergodic averages. To appear in *Transactions of the American Mathematical Society*.

[8] Jeremy Avigad and Richard Zach. The epsilon calculus. Stanford Encyclopedia of Philosophy, 2002.
http://plato.stanford.edu/entries/epsilon-calculus/.

[9] Ulrich Berger, Wilfried Buchholz, and Helmut Schwichtenberg. Refined program extraction from classical proofs. *Ann. Pure Appl. Logic*, 114:3–25, 2002.

[10] Ulrich Berger, Helmut Schwichtenberg, and Monika Seisenberger. The Warshall algorithm and Dickson's lemma: two examples of realistic program extraction. *Journal of Automated Reasoning*, 26:205–221, 2001.

[11] Wilfried Buchholz. Notation systems for infinitary derivations. *Archive for Mathematical Logic*, 30:277–296, 1991.

[12] Thierry Coquand and Martin Hofmann. A new method of establishing conservativity of classical systems over their intuitionistic version. *Mathematical Structures in Computer Science*, 9:323–333, 1999.

[13] Solomon Feferman. Infinity in mathematics: Is Cantor necessary? In G. Toraldo di Francia, editor, *L'infinito nella scienza*, pages 151–209. Istituto della Enciclopedia Italiana, 1987. Reprinted in Solomon Feferman, *In the Light of Logic*, Oxford University Press, New York, 1998, pages 28–73 and 229–248.

[14] Harvey M. Friedman. Classically and intuitionistically provable functions. In H. Müller and D. Scott, editors, *Higher Set Theory*, pages 21–27. Springer, Berlin, 1978.

[15] Gerhard Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112:493–465, 1936. Translated as "The consistency of elementary number theory" in Gerhard Gentzen, *Collected Works* (edited by M. E. Szabo), North-Holland, Amsterdam, 1969, pages 132–213.

[16] Jean-Yves Girard. *Proofs and Types,* translated and with appendices by Paul Taylor and Yves Lafont. Cambridge University Press, Cambridge, 1989.

[17] Jean-Yves Girard. A new constructive logic: classical logic. *Math. Structures Comput. Sci.*, 1:255–296, 1991.

[18] Kurt Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958. Reprinted with English translation in Feferman et al., eds., *Kurt Gödel: Collected Works*, volume 2, Oxford University Press, New York, 1990, pages 241–251.

[19] David Hilbert and Paul Bernays. *Grundlagen der Mathematik*. Springer, Berlin, first volume, 1934, second volume, 1939.

[20] J. Roger Hindley and Jonathan P. Seldin. *Introduction to Combinators and λ-calculus*. Cambridge University Press, Cambridge, 1986.

[21] C. A. R. Hoare. The verifying compiler: A grand challenge for computing research. *J. ACM*, 50:63–69, 2003.

[22] S. C. Kleene. On the interpretation of intuitionistic number theory. *J. Symbolic Logic*, 10:109–124, 1945.

[23] S. S. Kleene. *Introduction to Metamathematics*. North-Holland, Amsterdam, fourth reprint, 1964 edition, 1952.

[24] U. Kohlenbach. *Applied proof theory: proof interpretations and their use in mathematics*. Springer-Verlag, Berlin, 2008.

[25] Ulrich Kohlenbach. On the no-counterexample interpretation. *J. Symbolic Logic*, 64:1491–1511, 1999.

[26] Ulrich Kohlenbach and Paulo Oliva. Proof mining: a systematic way of analyzing proofs in mathematics. *Tr. Mat. Inst. Steklova*, 242(Mat. Logika i Algebra):147–175, 2003.

[27] Georg Kreisel. Interpretation of analysis by means of constructive functionals of finite type. In Arendt Heyting, editor, *Constructivity in Mathematics*, North-Holland, Amsterdam, 1959, pages 101–128.

[28] Grigori Mints (Minc). Quantifier-free and one-quantifier systems. *Zap. Naučn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 20:115–133, 285, 1971.

[29] Grigori Mints (Minc). What can be done in primitive recursive arithmetic. *Zap. Naučn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI)*, 60:93–102, 223–224, 1976. Studies in constructive mathematics and mathematical logic, VII.

[30] Grigori Mints. Proof theory in the USSR 1925–1969. *Journal of Symbolic Logic*, 56:385–424, 1991.

[31] Grigori Mints. *Selected Papers in Proof Theory*. Bibliopolis / North-Holland, Naples / Amsterdam, 1992.

[32] Grigori Mints. Normalization of finite terms and derivations via infinite ones. [31], pages 73–76.

[33] Grigori Mints. On E-theorems. [31], pages 105–115.

[34] Grigori Mints. Stability of E-theorems and program verification. [31], pages 117–121.

[35] Grigori Mints. Strong termination for the epsilon substitution method. *Journal of Symbolic Logic*, 61:1193–1205, 1996.

[36] Grigori Mints. Cut elimination for a simple formulation of epsilon calculus. *Ann. Pure Appl. Logic*, 152(1-3):148–160, 2008.

[37] Grigori Mints and Sergei Tupailo. Epsilon subsitution method for the ramified language and Δ^1_1-comprehension rule. In Andrea Cantini et al., editor, *Logic and Foundations of Mathematics*, pages 107–130. Kluwer, the Netherlands, 1999.

[38] Chetan R. Murthy. An evaluation semantics for classical proofs. In *Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 96–107, Amsterdam, 1991.

[39] Paulo Oliva. Unifying functional interpretations. *Notre Dame J. Formal Logic*, 47:263–290, 2006.

[40] Frank Pfenning. Logical frameworks. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 1063–1147. Elsevier and MIT Press, 2001.

[41] Helmut Schwichtenberg. Proof theory: Some aspects of cut-elimination. In Jon Barwise, editor, *Handbook of Mathematical Logic*, North-Holland, Amsterdam, 1977, pages 867–895.

[42] Joseph R. Shoenfield. *Mathematical Logic*. Association for Symbolic Logic, Urbana, IL, 2001. Reprint of the 1973 second printing.

[43] Stephen G. Simpson. *Subsystems of Second-Order Arithmetic*. Springer, Berlin, 1999.

[44] Thomas Streicher and Ulrich Kohlenbach. Shoenfield is Gödel after Krivine. *MLQ Math. Log. Q.*, 53:176–179, 2007.

[45] A. S. Troelstra. Realizability. In *Handbook of proof theory*, volume 137 of *Stud. Logic Found. Math.*, pages 407–473. North-Holland, Amsterdam, 1998.

[46] A. S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, Cambridge, second edition, 2000.

[47] Christian Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, 2000.

[48] Christian Urban and Gavin M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fund. Inform.*, 45:123–155, 2001.