Update procedures and the 1-consistency of arithmetic[*]

Jeremy Avigad

February 8, 2002

Abstract

The 1-consistency of arithmetic is shown to be equivalent to the existence of fixed points of a certain type of update procedure, which is implicit in the epsilon-substitution method.

1 Introduction

A theory in the language of arithmetic is said to be 1-consistent if it is consistent with every true universal sentence, or, equivalently, if every existential sentence it proves is true. The main theorem in this paper asserts that the 1-consistency of first-order Peano arithmetic is equivalent, over a weak metatheory, to the assertion that one can always solve certain systems of equations involving finite partial functions on the natural numbers.

It has been noted before (for example, by Tait, in [17]) that one can view Hilbert's epsilon-substitution method as posing the problem of finding solutions to systems of equations of a certain kind. The theorem just mentioned simply characterizes the types of equations that need to be solved, in a way that, I hope, is intuitive and helps clarify the computational content of the problem.

Lucid descriptions of epsilon-substitution method, with detailed references, can be found in [11] and [12]. In this paper, I will use Skolem function symbols instead of epsilon terms, more along the lines of [15] and [3]. The differences between the two ways of formulating the problem are minor, however, and are discussed below.

The outline of the paper is as follows. In Section 2, I describe the relevant systems of fixed-point equations, and use a short, but non-finitary, proof to show that the existence of solutions is guaranteed. This is similar to non-finitary proofs of the existence of solutions for the epsilon-substitution method in [12] and [16]. In Section 3, I show that, in a finitary metatheory, the existence of solutions to the systems of equations described implies the 1-consistency of

[*]Final version in *Mathematical Logic Quarterly*, 48(1):3–13, 2002. Work partially supported by NSF Grant DMS 0070600.

arithmetic. In Section 4, I provide a refined version of the existence proof, and show that solutions to the equations in question can be found using recursion on ordinals less than ε_0. The idea behind this proof is, in a sense, implicit in Ackermann's proof of termination for the epsilon-substitution method [1]. It is also implicit in Tait's approach [16, 17], which is based on similar continuity considerations.

Assuming one has enough basic functions in arithmetic so that every bounded formula is equivalent to one that is quantifier-free, or appealing to the MRDP theorem (see the discussion in [10, Section I.3.d]), the 1-consistency of arithmetic is equivalent to its Π_2 soundness. This is, in turn, equivalent to saying that every recursive function whose totality is provable in arithmetic is, in fact, total. Since PA proves that every $\prec\varepsilon_0$-recursive function is total, we have the desired equivalence of the 1-consistency of arithmetic and the fixed-point principle. In addition, we see that the provably total recursive functions of arithmetic can also be characterized as those functions that can be computed from fixed-point solutions to the systems of equations we have described.

With these facts in mind, the main theorem of this paper can be seen as a small contribution to the general proof-theoretic program of trying to characterize the assertions of consistency or 1-consistency of various theories, in combinatorial or computational terms. This program is currently being pursued vigorously by Harvey Friedman (see, for example, [8], or [9] and the references at the end). Friedman's results are more dramatic than the ones presented here, in two respects: first, the theories he considers are much stronger than arithmetic; and second, he has gone further in eliminating all traces of logical and metamathematical notions from the principles he obtains. Nonetheless, the fixed-point principle discussed here is interesting insofar as it is fairly natural and simple. It is worth exploring whether or not this principle can be couched in more common mathematical terms.

As in [3], the methods extend immediately to extensions of arithmetic with transfinite induction principles. Using ideas from [2, 3, 13] it should be possible to characterize the 1-consistency of predicative analysis in terms of fixed points of certain transfinite sets of equations. With ideas from [4], it should be possible to extend this to impredicative theories like Kripke-Platek set theory as well. Here, too, it remains to be seen whether principles obtained in this way can be made mathematically intuitive or compelling.

This paper grew out of a discussion with Grigori Mints on epsilon substitution, and I am grateful to him for the conversation. I am also grateful to William Tait for comments and suggestions, and especially for a simplification in the proofs of Theorems 2.2 and 4.4.

2 Update procedures

A *finite partial function from* $\mathbb{N}$ *to* $\mathbb{N}$ is a function from some finite subset of the natural numbers to the set of natural numbers. Below, the phrase "finite partial function" always means "finite partial function from $\mathbb{N}$ to $\mathbb{N}$," and the variables

ρ, σ, τ range over such functions. I will write $\sigma \supseteq \tau$ to denote that σ extends τ, i.e. the domain of σ includes that of τ and the two functions agree on the domain of τ. Also, I will use $\emptyset$ to denote the partial function that is nowhere defined. If σ is a finite partial function, I will use $\hat{\sigma}$ to denote the extension of σ to a total function, defined by

$$\hat{\sigma}(x) = \begin{cases} \sigma(x) & \text{if } x \text{ is in the domain of } \sigma \\ 0 & \text{otherwise.} \end{cases}$$

Given a pair $\langle u, v \rangle$ of natural numbers, let $\sigma \oplus \langle u, v \rangle$ to denote the modification of σ that maps u to v, and agrees with σ otherwise; in other words,

$$(\sigma \oplus \langle u, v \rangle)(x) = \begin{cases} \sigma(x) & \text{if } x \text{ is in the domain of } \sigma \text{, and } x \neq u \\ v & \text{if } x = u \\ \text{undefined} & \text{otherwise.} \end{cases}$$

It will be convenient to extend the $\oplus$ operation to a "null" value $\emptyset$, defining $\sigma \oplus \emptyset$ to be σ.

Let $F(f_1, \ldots, f_k)$ be a functional, mapping (total) functions $f_1, \ldots, f_k$ to some set S. Say that F is *continuous* if it is continuous with respect to the usual product topology on $\mathbb{N}^{\mathbb{N}}$, assuming S is given the discrete topology. In other words, F is continuous if for every $f_1, \ldots, f_k$ there is a finite set of natural numbers A, such that whenever functions $g_1, \ldots, g_k$ agree with $f_1, \ldots, f_k$, respectively, on A, then $F(f_1, \ldots, f_k) = F(g_1, \ldots, g_k)$. Stated less formally, F is continuous if the value of $F(f_1, \ldots, f_k)$ depends on only finitely many values of the functions $f_1, \ldots, f_k$.

Now, suppose $F(g, f_1, \ldots, f_k)$ is a continuous functional with range $\mathbb{N} \times \mathbb{N} \cup \{\emptyset\}$, and, for a given sequence $f_1, \ldots, f_k$, consider the act of replacing a finite partial function σ by $\sigma \oplus F(\hat{\sigma}, f_1, \ldots, f_k)$. Say that F is an *update procedure* in g if the following holds: whenever $F(\hat{\sigma}, f_1, \ldots, f_k) = \langle a, b \rangle$, τ extends $\sigma \oplus \langle a, b \rangle$, and $F(\hat{\tau}, h_1, \ldots, h_k) = \langle a, c \rangle$, then $b = c$. In other words, once F "sets" the value of $\sigma(a)$ to b, it does not change it, no matter how the other arguments vary.

If $F(g)$ is a unary update procedure, a *finite fixed point* of F is a finite partial function σ such that $\sigma = \sigma \oplus F(\hat{\sigma})$.

Lemma 2.1 *Every unary update procedure has a finite fixed point. Moreover, suppose $H(g, \vec{h})$ is a continuous functional and for each fixed choice of $\vec{h}$ the functional $g \mapsto H(g, \vec{h})$ is an update procedure. Then there is a continuous functional $G(\vec{h})$ such that for every $\vec{h}$, $G(\vec{h})$ is a finite fixed point of the functional $g \mapsto H(g, \vec{h})$.*

Proof. Suppose $H(g, \vec{h})$ is continuous, and $g \mapsto H(g, \vec{h})$ is an update procedure for each $\vec{h}$. Fix $\vec{h}$ for the moment, and define a sequence $\sigma^0, \sigma^1, \sigma^2, \ldots$ by letting $\sigma^0 = \emptyset$ and, for each i, letting $\sigma^{i+1} = \sigma^i \oplus H(\hat{\sigma}^i, \vec{h})$. The crucial point is that this is an increasing sequence: the fact that $g \mapsto H(g, \vec{h})$ is an update procedure implies that $\sigma^0 \subseteq \sigma^1 \subseteq \ldots$. Let g be the partial function extending all of the σ^i, that is, $g = \bigcup_{i \in \mathbb{N}} \sigma^i$. The continuity of H implies that for some i, we have $H(\hat{g}, \vec{h}) = H(\hat{\sigma}^i, \vec{h}) = H(\hat{\sigma}^{i+1}, \vec{h}) = \ldots$. But then $\sigma^{i+1} = \sigma^i \oplus H(\hat{\sigma}^i, \vec{h}) = \sigma^{i+1} \oplus H(\hat{\sigma}^i, \vec{h})$, so σ^{i+1} is the desired fixed point.

The continuity of H implies that only finitely many values of $\vec{h}$ are used in the computation of $\sigma^0, \sigma^1, \ldots, \sigma^{i+1}$. Hence, if we let $G(\vec{h})$ denote the fixed point obtained by the procedure above, G is a continuous functional. □

A *system of nested update procedures* is a sequence of continuous functionals $F_1(f_1, \ldots, f_n), \ldots, F_n(f_1, \ldots, f_n)$ such that for each i and fixed $f_1, \ldots, f_{i-1}$, the functional

$$f_i, f_{i+1}, \ldots, f_n \mapsto F_i(f_1, \ldots, f_n)$$

is an update procedure for f_i. Note that additional arguments are held fixed as one proceeds from 1 to n, so the order is important. A *finite fixed point* of such a system is a sequence of finite partial functions $\sigma_1, \ldots, \sigma_n$ such that the equations

$$\begin{aligned} \sigma_1 &= \sigma_1 \oplus F_1(\hat{\sigma}_1, \ldots, \hat{\sigma}_n) \\ \vdots & \quad \vdots \\ \sigma_n &= \sigma_n \oplus F_n(\hat{\sigma}_1, \ldots, \hat{\sigma}_n) \end{aligned}$$

are all satisfied.

Theorem 2.2 *Every system of nested update procedures has a finite fixed point.*

Proof. We will prove this by induction on n. The case where $n = 1$ is just Lemma 2.1.

For the induction step, suppose $F_1(f_1, \ldots, f_{n+1}), \ldots, F_{n+1}(f_1, \ldots, f_{n+1})$ is a system of nested update procedures of size $n + 1$. Using Lemma 2.1, let $G(f_1, \ldots, f_n)$ be a continuous functional returning finite fixed points of the functionals $f_{n+1} \mapsto F_{n+1}(f_1, \ldots, f_{n+1})$. It is not difficult to verify that the sequence of functionals

$$f_1, \ldots, f_n \mapsto F_i(f_1, \ldots, f_n, G(f_1, \ldots, f_n))$$

for $i = 1, \ldots, n$ is a system of nested update procedures of size n. (We know that for each i and fixed $f_1, \ldots, f_{i-1}$, the functional $f_i, \ldots, f_{n+1} \mapsto F_i(f_1, \ldots, f_{n+1})$ is an update procedure for f_i; but this means that the appropriate monotonicity condition holds, no matter how f_{n+1} varies.) By the induction hypothesis, the smaller system has a finite fixed point, $\sigma_1, \ldots, \sigma_n$. Letting $\sigma_{n+1} = G(\hat{\sigma}_1, \ldots, \hat{\sigma}_n)$ yields a finite fixed point of the larger system. □

The statement of Theorem 2.2 is hardly finitary, since it quantifies over arbitrary functionals. But the proof is *is* constructive, if one takes the continuous functionals to be Brouwer functionals, meaning that the set of unsecured sequences is well-founded. Lemma 4.3 below, as well as the developments in [16, 17], is essentially a refinement of this observation.

We will obtain an assertion equivalent to the 1-consistency of *PA* by restricting the set of functionals we consider. There are a number of ways to do this; one way is to restrict our attention to functionals that are *elementary.*

The set of elementary functions is a set of functions from the natural numbers to the natural numbers of various arities, where a 0-ary function is just a constant; it can be defined as the smallest set of functions containing zero, projections, successor, addition, multiplication, and exponentiation, and closed under composition and bounded recursion. Similarly, the set of elementary functions in $f_1, \ldots, f_k$ is the smallest set satisfying these conditions and containing also $f_1, \ldots, f_k$. One can view an l-ary elementary function in $f_1, \ldots, f_k$ as a functional $F(x_1, \ldots, x_l, f_1, \ldots, f_k)$, and any such functional is necessarily continuous. If $l = 0$, F has no numeric parameters, and one has a functional of the kind considered above.

For the rest of this paper, the reader only needs to accept that elementary functions are strong enough to carry out straightforward computations and syntactic operations, and that one can develop an adequate theory of the elementary functions in primitive recursive arithmetic, *PRA*. For more details, the reader is referred to [3, 14].

A *system of nested elementary update procedures* is a system of nested update procedures in which all the functionals are elementary. The main theorem in this paper is the following:

Theorem 2.3 *PRA proves that the following are pairwise equivalent:*

1. *Every* $\prec\varepsilon_0$*-recursive function is total.*
2. *Every system of nested elementary update procedures has a finite fixed point.*
3. *Every* Π_2 *sentence provable in PA is true.*

A definition of the $\prec\varepsilon_0$-recursive functions will be provided below. The proof that 3 implies 1 in *PRA* is fairly standard: for each α less than ε_0, one can prove instances of transfinite induction up to α in *PA*, and use that to show that every α-recursive function is total (see, for example, [6]). Since this last statement is Π_2, we have the desired conclusion. In Section 3, I will show that *PRA* proves that 1 implies 2, and in Section 4, I will show that *PRA* proves that 2 implies 3. Of course, the fact that *PRA* proves the equivalence of 1 and 3 is well known; so what is new here is the equivalence of 2, and the resulting proof that 1 implies 3.

There is nothing special about the choice of elementary functionals: we only need a collection of functionals that is restricted enough so that we can quantify over them in our metatheory, yet expressive enough so that we can carry out the basic syntactic operations of Section 3. Also, with our current setup, we can get by with a metatheory weaker than *PRA*; theories asserting the totality of iterated exponentiation like $I\Delta_0(superexp)$ or EFA^* suffice (see [10]). Finally, with some additional work, we can also characterize the strength of the fragments of arithmetic $I\Sigma_n$ in terms of nested systems of equations of depth n. More information can be found in [3].

3 The 1-consistency of arithmetic

In this section I will show that the existence of finite fixed points of systems of nested elementary update procedures implies the 1-consistency of arithmetic. All the definitions, theorems, and proofs in this section and the next should be thought of as taking place in *PRA*.

I will take the language of arithmetic to include symbols for all the elementary functions and relations. Peano arithmetic then consists of quantifier-free axioms for the basic function and relation symbols, together with a schema of induction for arbitrary formulae in the language. The advantage to including elementary functions among the basic symbols is that now, as noted in the introduction, the Π_2 soundness of arithmetic is easily shown to be equivalent to its 1-consistency.

We would like to embed *PA* in a quantifier-free calculus. To that end, we can use either epsilon terms or Skolem functions to name least witnesses to existential formulae. The main difference is that in the epsilon calculus, terms can be read ambiguously; for example, if x and y are not free in s and x is not free in η, one can read a term

$$\varepsilon_x \theta(x, \varepsilon_y \eta(y, s))$$

as either the result of applying the "function" $\lambda w\ \varepsilon_x \theta(x, w)$ to $\varepsilon_y \eta(y, s)$, or as applying the "function" $\lambda u\ \varepsilon_x \theta(x, \varepsilon_y \eta(y, u))$ to s. We will use Skolem functions instead, which forces one to distinguish between the two, thereby simplifying some of the definitions and proofs below. Epsilon terms have the advantage of allowing one to interpret quantifiers directly, and that would have enabled us to avoid the use of Herbrand's theorem in the proof of Lemma 3.2. Alternatively, one can interpret epsilon terms as Skolem functions in a canonical way, as in [12, 17]. In the end, the differences between the two approaches are minor.

The lemmata below provide, essentially, an "abstract" version of the usual epsilon substitution method; compare, for example, the proof of Lemma 3.3 to the presentation in [12].

Our first task, then, is to use Skolem functions to embed *PA* in a quantifier-free theory. Let L_0 be the language of arithmetic, and recursively let L_{i+1} denote the language that results from adding to L_i a new function symbol $\mu_\theta(\vec{y})$ for each quantifier-free formula $\theta(x, \vec{y})$ in L_i with the distinct free variables shown. Let L_ω denote the union of the L_i. Let PA^h be the theory in the language L_ω axiomatized by the quantifier-free defining axioms for the elementary functions and relations, together with axioms

$$\theta(x, \vec{y}) \to \theta(\mu_\theta(\vec{y}), \vec{y}) \land \mu_\theta(\vec{y}) \leq x. \tag{1}$$

The *rank* of a symbol μ_θ is the least i such that μ_θ is in L_i.

Using the new axioms, it is not difficult to show that in PA^h every formula in the language of L_ω is equivalent to a quantifier-free one, and the schema of induction then follows from (1). As a result, *PA* is contained in PA^h.

Let S be a finite set of closed instances of the μ-axioms involving only the μ-symbols $\mu_1, \mu_2, \ldots, \mu_n$. Given a sequence of unary functions $f_1, \ldots, f_n$, we can interpret each symbol $\mu_i(\vec{y})$ by the function $\vec{y} \mapsto f_i(\langle \vec{y} \rangle)$, assuming that we have chosen a reasonable (elementary) coding of sequences $\langle \vec{y} \rangle$ of natural numbers by a single natural number. If t is a term mentioned in S, let $t^{\vec{f}}$ denote its value under $\vec{f}$, interpreting the function symbols of L_0 by the elementary functions they are intended to denote. Since every sentence in S is quantifier-free, for each sentence φ we can compute its truth value $\varphi^{\vec{f}}$ under this interpretation of the terms, interpreting the relation symbols of L_0 in the expected way.

Definition 3.1 *Let S be a finite set of closed instances of μ-axioms involving only the μ-symbols $\mu_1, \ldots, \mu_n$. Let $\sigma_1, \ldots, \sigma_n$ be a sequence of finite partial functions. The sequence $\sigma_1, \ldots, \sigma_n$ is a* finite partial model *of S if every formula in S is true under the interpretation $\hat{\sigma}_1, \ldots, \hat{\sigma}_n$.*

Lemma 3.2 *Suppose every finite set of closed instances of μ-axioms has a finite partial model. Then PA is 1-consistent.*

Proof. Suppose every finite set of closed instances of μ-axioms has a finite partial model, and suppose PA proves $\exists x_1, \ldots, x_k\ \theta(x_1, \ldots, x_k)$, where $\theta(x_1, \ldots, x_k)$ is quantifier-free with at most the free variables shown. Then PA^h proves $\theta(t_1, \ldots, t_k)$, where $t_1, \ldots, t_k$ are appropriate closed terms involving the μ-symbols. By Herbrand's theorem, there is a propositional proof of $\theta(t_1, \ldots, t_k)$ from closed instances of the equality axioms and axioms of PA^h. Let S be the set of instances of μ-axioms used in this propositional proof, and let $\vec{\sigma}$ be a finite partial model of S. The equality axioms are clearly true when interpreted under $\vec{\sigma}$, and instances of the axioms for the elementary functions and relations are true because they are interpreted in the standard way. (We can interpret μ-symbols not appearing in S by the constant zero function.) Since all the axioms used in the propositional proof are true under $\vec{\sigma}$, the conclusion is also true under $\vec{\sigma}$. Then θ holds of the interpretation of $t_1, \ldots, t_k$, and so $\exists x_1, \ldots, x_k\ \theta(x_1, \ldots, x_k)$ is true. □

The converse of the statement of Lemma 3.2 is also provable in PRA, but we will not need this fact below. To finish proving the first implication of Theorem 2.3, we only need to prove the following lemma.

Lemma 3.3 *Suppose every system of nested elementary update procedures has a finite fixed point. Then every set of closed instances of μ-axioms has a finite partial model.*

Proof. Fix a finite set S of closed instances of μ-axioms. We will design a system of nested elementary update procedures, such that a finite fixed point of the system is a finite partial model of S.

Suppose that $\mu_1, \ldots, \mu_n$ are the μ-symbols mentioned in S. By reordering them if necessary, we can assume that whenever i is less than j, the rank of μ_i is less than or equal to the rank of μ_j. This means that if $\mu_i(\vec{y})$ is intended to denote the least element function for $\theta(x, \vec{y})$ according to axiom (1) above, then at most the μ-symbols $\mu_1, \ldots, \mu_{i-1}$ occur in θ.

For each i, define $F_i(f_1, \ldots, f_n)$ as follows. If μ_i has defining axiom (1), then S contains finitely many closed substitution instances of this axiom, and these are of the form

$$\theta(s, t_1, \ldots, t_k) \rightarrow \theta(\mu_i(t_1, \ldots, t_k), t_1, \ldots, t_k) \wedge \mu_i(t_1, \ldots, t_k) \leq s. \qquad (2)$$

We can assume that the sentences occurring in S are ordered in some canonical way, say, by their Gödel numbers. If every instance of (2) in S is true under $\vec{f}$, let $F_i(f_1, \ldots, f_n)$ return the null update, $\emptyset$. Otherwise, choose the first instance of (2) that is false. In that case, we know that $\theta^{\vec{f}}(s^{\vec{f}}, t_1^{\vec{f}}, \ldots, t_k^{\vec{f}})$ is true, but either $\theta^{\vec{f}}(f(t_1^{\vec{f}}, \ldots, t_k^{\vec{f}}), t_1^{\vec{f}}, \ldots, t_k^{\vec{f}})$ is false or $s^{\vec{f}}$ is less than $f(t_1^{\vec{f}}, \ldots, t_k^{\vec{f}})$. Let m be the least natural number less than or equal to $s^{\vec{f}}$ such that $\theta^f(m, t_1^{\vec{f}}, \ldots, t_k^{\vec{f}})$ is true, and let $F_i(f_1, \ldots, f_n)$ return the update $\langle\langle t_1^{\vec{f}}, \ldots, t_k^{\vec{f}}\rangle, m\rangle$.

There are only two things to check. First, we need to confirm that for each i and fixed $f_1, \ldots, f_{i-1}$, the functional $f_i, \ldots, f_n \mapsto F_i(f_1, \ldots, f_n)$ is an update procedure. But this is true since $F_i(f_1, \ldots, f_n)$ outputs an update $\langle a, b\rangle$ only if b is the least value making $\theta^{\vec{f}}(b, (a)_0, \ldots, (a)_k)$ true, and this determination does not change if we vary $f_i, \ldots, f_n$, since $\mu_i, \ldots, \mu_n$ do not appear in $\theta(x, \vec{y})$.

Second, we need to confirm that if $\sigma_1, \ldots, \sigma_n$ is a finite fixed point of the system of update procedures, then it is a finite partial model of S. But this is also clear: if an instance (2) of an axiom for μ_i is false under the interpretation $\hat{\sigma}_1, \ldots, \hat{\sigma}_k$, then $\sigma_i \oplus F_i(\hat{\sigma}_1, \ldots, \hat{\sigma}_n)$ is different from σ_i. $\square$

It may be worth noting that the lemma does not depend on the interpretation of μ-symbols in terms of *least* witnesses, so these methods can also be used for theories with Skolem functions that return witnesses that are not unique.

4 Ordinal recursion

Theorem 2.2 asserted that every system of nested update procedures has a fixed point. In this section, I will present a more constructive proof, for the case where the functionals involved are elementary. First, I will characterize the $\prec\varepsilon_0$-recursive functionals as set of functionals that can be computed using an iterative procedure that counts down through ordinals below ε_0. Then I will show that for the systems of equations under considerations, the fixed points can be computed by such functions.

Classically, ε_0 can be defined as the least fixed point of the function $\alpha \mapsto \omega^\alpha$. Equivalently, it is the limit of the sequence $\langle\omega_n\rangle_{n\in\omega}$, where $\omega_0 = 1$ and $\omega_{n+1} = \omega^{\omega_n}$, and the least ordinal closed under addition and exponentiation. I will assume that we have fixed an elementary well-ordering $\prec$ of order-type ε_0, with elementary operations mirroring the usual functions of addition, exponentiation, etc. on ordinals, such that basic properties can be proved in *PRA*. One can make this more precise by assuming that we have an *elementary recursive ordinal notation system* for ε_0, in the terminology of [7]; a list of the "usual properties" the functions are are to satisfy appears in [7, Section 1].

To interpret the following definition as taking place in PRA, one should take the variables $f_1, \ldots, f_k$ to range over elementary functions (via their codes). A *α-recursive* functional $F(x_1, \ldots, x_l, f_1, \ldots, f_k)$ is given by a notation α and elementary functions $\mathit{start}(x_1, \ldots, x_l)$, $\mathit{next}(q, u_1, \ldots, u_k)$, $\mathit{query}_1(q), \ldots, \mathit{query}_k(q)$, $\mathit{norm}(q)$, and $\mathit{result}(q)$. Informally, these data describe the functional whose values are computed in the following way: on input $x_1, \ldots, x_l$, the algorithm begins in state $\mathit{start}(x_1, \ldots, x_l)$. As long as the norm of the current state q is less than α and the norm of the previous state, the algorithm queries the functions $f_1, \ldots, f_k$ at $\mathit{query}_1(q), \ldots, \mathit{query}_k(q)$, respectively, and, based on the responses $u_1, \ldots, u_k$ to these queries and the current state, proceeds to the next state, $\mathit{next}(q, u_1, \ldots, u_k)$. When the norm of the state q is not less than the norm of the previous state, the computation returns $\mathit{result}(q)$.

More formally, s is a *computation sequence* for F at the values $\vec{x}, \vec{f}$ if s is a sequence $\langle s_0, s_1, s_2, \ldots, s_m \rangle$ satisfying the following: $s_0 = \mathit{start}(\vec{x})$; for every $i < m$, $s_{i+1} = \mathit{next}(s_i, f_1(\mathit{query}_1(s_i)), \ldots, f_k(\mathit{query}_k(s_i)))$; and either $m = 0$ and $\mathit{norm}(s_0) \not\prec \alpha$, or $m > 0$, $\mathit{norm}(s_0) \prec \alpha$, $\mathit{norm}(s_{i+1}) \prec \mathit{norm}(s_i)$ for every $i < m - 1$, and $\mathit{norm}(s_m) \not\prec \mathit{norm}(s_{m-1})$. F is *defined* at $\vec{x}, \vec{f}$ if there is a computation sequence s for F at $\vec{x}, \vec{f}$, and in that case, the value of $F(\vec{x}, \vec{f})$ is said to be $\mathit{result}(s_m)$, where s_m is the last element of s.

A functional is said to be *$\prec\varepsilon_0$-recursive* if it is α-recursive for some $\alpha \prec \varepsilon_0$. If $k = 0$ in the example above, $F(x_1, \ldots, x_l)$ is a $\prec\varepsilon_0$-recursive *function*, so we can now take the statement "every $\prec\varepsilon_0$-recursive function is total" to mean that for every such F and $x_1, \ldots, x_l$, $F(x_1, \ldots, x_l)$ is defined. Even in the special case where l is also zero, the general statement that every ε_0-recursively specified *value* F is defined is nontrivial (and is, in fact, equivalent to corresponding assertion for functions). In [3] it is shown, for example, that the set of $\prec\varepsilon_0$-recursive functions is closed under composition.

Some basic properties of this model of recursion are described in [3]. In particular, we will need the following:

Lemma 4.1 *Suppose $F(x_1, \ldots, x_l, f_1, \ldots, f_k)$ is an elementary recursive (or even primitive recursive) functional. Then it is $\prec\omega^\omega$-recursive.*

Lemma 4.2 *Suppose α is infinite and closed under addition. Then the set of $\prec\alpha$-recursive functionals is closed under composition.*

In particular, set of the $\prec\varepsilon_0$-recursive functionals includes the set of elementary recursive ones, and is closed under composition.

The following is our constructive analogue of Lemma 2.1. The idea behind the proof is also used in [5] and [3]; see also [16, Section 7].

Lemma 4.3 *Suppose $H(g, h_1, \ldots, h_l)$ is α-recursive, and for each fixed sequence of elementary functions $\vec{h}$, the functional $g \mapsto H(g, \vec{h})$ is an update procedure.*

Then there is an ω^α-recursive functional $G(\vec{h})$, such that for each $\vec{h}$, if $G(\vec{h})$ is defined, then it is a finite fixed point of $g \mapsto H(g, \vec{h})$.

Proof. The proof has the flavor of a finite injury priority argument. Given $\vec{h}$, the idea is as follows. Start by setting $\sigma^0 = \emptyset$, the first approximation to a fixed point, and then carry out the computation of H at $\hat{\sigma}^0, \vec{h}$. This yields a computation sequence $s_0, s_1, \ldots, s_m$ and a descending sequence of ordinals $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$. If the computation yields a pair $\langle u, v\rangle$, we must update σ^0 by setting the value of the function at u equal to v. This may invalidate the computation sequence, say at stage i, if the computation of $H(\hat{\sigma}_0, \vec{h})$ queried the value of $\hat{\sigma}^0(u)$ at that stage. In that case, we have to throw away the rest of the computation sequence, and begin anew from stage i with updated value of $\hat{\sigma}(u)$. But some progress has been made: we know we will never have to change this value again. Of course, a later update might invalidate the computation sequence at an even earlier stage, say at stage $j < i$; but then even greater progress will have been made. An appropriate assignment of ordinals below ω^α guarantees that this procedure will terminate. The details follow.

Given $H(g, \vec{h})$, we need to describe an algorithm to compute $G(\vec{h})$. Say a *partial computation sequence* of H at $g, \vec{h}$ is a proper initial segment of a computation sequence, i.e. a sequence satisfying the definition above except that the norm of the last state is less than the norm of the previous one. States of G will be pairs of the form $\langle \sigma, \langle s_0, \ldots, s_m\rangle\rangle$, where σ is a finite partial function and $\langle s_0, \ldots, s_m\rangle$ is a partial computation sequence for $H(\hat{\sigma}, \vec{h})$. The starting state will be $\langle \emptyset, \mathit{start}_H\rangle$, and the starting ordinal will be ω^α.

If q is the state $\langle \sigma, \langle s_0, \ldots, s_m\rangle\rangle$, assign a norm to q as follows. For each i such that $0 \le i \le m$, let

$$\delta_i = \begin{cases} 2 & \text{if } \mathit{query}_H(s_i) \text{ is not in the domain of } \sigma \\ 0 & \text{otherwise.} \end{cases}$$

Then define $\mathit{norm}(q)$ to be

$$\omega^{\mathit{norm}(s_0)} \cdot \delta_0 + \ldots + \omega^{\mathit{norm}(s_{m-1})} \cdot \delta_{m-1} + \omega^{\mathit{norm}(s_m)} \cdot (\delta_m + 1).$$

Finally, in state q, determine the next state, $\mathit{next}(q)$, as follows. First, compute $s_{m+1} = \mathit{next}_H(s_m, \hat{\sigma}(\mathit{query}_{H,1}(s_m)), g_1(\mathit{query}_{H,2}(s_m)), \ldots, g_l(\mathit{query}_{H,l}(s_m)))$. The value of $\mathit{next}(q)$ is defined by cases, as follows:

1. If $\mathit{norm}_H(s_{m+1}) \prec \mathit{norm}_H(s_m)$, the computation of $H(\hat{\sigma}, \vec{h})$ is not finished. Set $\mathit{next}(q) = \langle \sigma, \langle s_0, \ldots, s_m, s_{m+1}\rangle\rangle$. Note that the norm of this state is

   $$\omega^{\mathit{norm}(s_0)} \cdot \delta_0 + \ldots + \omega^{\mathit{norm}(s_m)} \cdot \delta_m + \omega^{\mathit{norm}(s_{m+1})} \cdot (\delta_{m+1} + 1),$$

   which is less than $\mathit{norm}(q)$.

2. Otherwise, $\mathit{norm}_H(s_{m+1}) \not\prec \mathit{norm}_H(s_m)$, and the computation of $H(\hat{\sigma}, \vec{h})$ is complete. Let $w = \mathit{result}_H(s_{m+1})$, and let $\sigma' = \sigma \oplus w$. Check to see if $\langle s_0, \ldots, s_{m+1}\rangle$ is also a correct computation sequence of H at $\hat{\sigma}', \vec{h}$.

(a) If it is not, then w must be of the form $\langle u, v\rangle$, where for some i, $\mathit{query}_{H,1}(s_i) = u$. Since H is an update procedure and the value of $\hat{\sigma}(u)$ is different from the value of $\hat{\sigma}'(u)$, we know that u is not in the domain of σ. Let i be the least value such that $\mathit{query}_{H,1}(s_i) = u$, and let $\mathit{next}(q)$ be $\langle \sigma', \langle s_0, \ldots, s_i\rangle\rangle$. Note that δ_i has dropped from 2 to 0, and so the norm of this state is

$$\omega^{\mathit{norm}(s_0)} \cdot \delta_0 + \ldots + \omega^{\mathit{norm}(s_{i-1})} \cdot \delta_{i-1} + \omega^{\mathit{norm}(s_i)} \cdot 1$$

which is less than $\mathit{norm}(q)$.

(b) Otherwise, $\sigma' \oplus w = \sigma' \oplus H(\hat{\sigma}', \vec{h})$. In that case, σ' is the desired fixed point and we are done. Let $\mathit{next}(q)$ code this fact, and let $\mathit{result}(\mathit{next}(q))$ return σ'.

It is not hard to verify that at each intermediate state $\langle \sigma, \langle s_0, \ldots, s_m\rangle\rangle$ of the computation, $\langle s_0, \ldots, s_m\rangle$ is a partial computation sequence for H at $\hat{\sigma}, \vec{g}$. The procedure terminates only when one has a value of σ and a computation sequence for $H(\hat{\sigma}, \vec{g})$ such that $\sigma = \sigma \oplus H(\hat{\sigma}, \vec{g})$. □

The following theorem is the constructive analogue of Theorem 2.2. The idea behind the proof is essentially the same, but here we have to be careful about the wording, to ensure that the proof can be carried out in *PRA*. We will use the fact that in proving Lemma 4.3, one can find a primitive recursive function that computes (the code for) G uniformly from (the code for) H.

Theorem 4.4 *Suppose every $\prec\varepsilon_0$-recursive function is total. Then every system of nested elementary update procedures has a finite fixed point.*

Proof. By recursion from n down to 1, let $G_n(f_1, \ldots, f_{n-1})$ return fixed points of the functionals $f_n \mapsto F_n(f_1, \ldots, f_n)$ as in Lemma 4.3; let $G_{n-1}(f_1, \ldots, f_{n-2})$ return fixed points of the functionals $f_{n-1} \mapsto F_{n-1}(f_1, \ldots, f_{n-1}, G_n(f_1, \ldots, f_{n-1}))$, and so on. In the end, consider the sequence

$$G_1, G_2(\hat{G}_1), G_3(\hat{G}_1, \widehat{G_2(\hat{G}_1)}), \ldots, G_n(\hat{G}_1, \widehat{G_2(\hat{G}_1)}, \ldots, \widehat{G_{n-1}(\ldots)}).$$

Since every $\prec\varepsilon_0$-recursive function is total, all these values are defined,[1] and provide the desired fixed point. □

References

[1] Wilhelm Ackermann. Zur Widerspruchsfreiheit der Zahlentheorie. *Mathematische Annalen*, 93:1–36, 1940.

[1]To be more precise: using the methods of [3] we can combine these values into a single $\prec\varepsilon_0$-recursive value K, such that if K is defined then all these other values are defined as well.

[2] Toshiyasu Arai. Epsilon substitution method for theories of jump hierarchies. Preprint.

[3] Jeremy Avigad. Ordinal analysis without proofs. To appear in *Reflections: A collection honoring Solomon Feferman on his 70th birthday*, ASL Lecture Notes in Logic, AK Peters.

[4] Jeremy Avigad. An ordinal analysis of admissible set theory using recursion on ordinal notations. Submitted.

[5] Samuel Buss. The witness function method and provably recursive functions of Peano arithmetic. In D. Westerståhl D. Prawitz, B. Skyrms, editor, *Proceedings of the Ninth International Congress on Logic, Methodology, and Philosophy of Science*, pages 29–68. Elsevier North-Holland, 1994.

[6] Matt Fairtlough and Stanley Wainer. Hierarchies of provably recursive functions. In Samuel Buss, editor, *The Handbook of Proof Theory*, North-Holland, 1998, pages 149–207.

[7] Harvey Friedman and Michael Sheard. Elementary descent recersion and proof theory. *Annals of Pure and Applied Logic*, 71:1–45, 1995.

[8] Harvey Friedman. Finite functions and the necessary use of large cardinals. *Annals of Mathematics*, 148:803–893, 1998.

[9] Harvey Friedman. Boolean relation theory. Posting to the Foundations of Mathematics forum, http://www.math.psu.edu/simpson/fom, March 10, 2000.

[10] Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic.* Springer, Berlin, 1993.

[11] Grigori Mints. Gentzen-type systems and Hilbert's epsilon subsitution method I. In D. Prawitz, B. Skyrms, and D. Westerståhl, editors, *Logic, Methodology, and Philosophy of Science IX*, pages 91–122. Elsevier Science, 1994.

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

[13] 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, 1999.

[14] H. E. Rose. *Subrecursion: Functions and Hierarchies.* Clarendon Press, 1984.

[15] Richard Sommer. *Trasnfinite Induction and Hierarchies Generated by Transfinite Recursion within Peano Arithmetic.* PhD thesis, University of California, Berkeley, 1990.

[16] William Tait. Functionals defined by transfinite recursion. *Journal of Symbolic Logic*, 30:155–174, 1965.

[17] William Tait. The substitution method. *Journal of Symbolic Logic*, 30:175–192, 1965.