Transfinite Update Procedures for Predicative Systems of Analysis

Federico Aschieri

Dipartimento di Informatica, Università di Torino
Italy
School of EECS, Queen Mary, University of London
United Kingdom

Abstract

We present a simple-to-state, abstract computational problem, whose solution implies the 1-consistency of various systems of predicative Analysis and offers a way of extracting witnesses from classical proofs. In order to state the problem, we formulate the concept of transfinite update procedure, which extends Avigad's notion of update procedure to the transfinite and can be seen as an axiomatization of learning as it implicitly appears in various computational interpretations of predicative Analysis. We give iterative and bar recursive solutions to the problem.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Update procedure, epsilon substitution method, predicative classical analysis, bar recursion

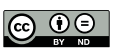
Digital Object Identifier 10.4230/LIPIcs.CSL.2011.20

1 Introduction

The aim of this paper is to provide an abstract description of learning as it is appears in various computational interpretations of predicative fragments of classical second order Arithmetic. Our account has a twofold motivation and interest.

Its first purpose is to provide a foundation that will serve to extend Aschieri and Berardi's learning based realizability for Heyting Arithmetic with EM_1 (see [3]) to predicative fragments of Analysis: a possible path to follow is the one suggested here. In particular, we describe and prove the termination of the learning processes that should arise when extending the approach of learning based realizability to predicative Arithmetic.

Secondly, we continue the work of Avigad on update procedures [4] and extend it to the transfinite case by introducing the concept of transfinite update procedure. The concept may be seen as an axiomatization of learning as implicitly used in the epsilon substitution method for Elementary Analysis and Ramified Analysis as formulated in the work of Mints et al. ([8], [9]). The idea is that one can associate with any classical proof in those systems of any formula $\exists x^{\mathbb{N}} P(x)$, with P computable, a transfinite update procedure. Each transfinite update procedure has a so called *zero*, representing a finite approximation of some transfinite sequence of oracles thanks to which it is possible to compute a n such that $P(n)$ holds. The problem is both to formulate efficient learning processes that build step by step this finite zero and to prove their termination. The notion of update procedure is useful to understand the combinatorial content and the fundamental ideas of the epsilon method in a totally abstract way. By formulating an abstract self-contained concept, we also hope to present to non-specialists a very challenging computational problem, whose efficient solution is of great importance for program extraction purposes in the proof theory of classical logic.



Computer Science Logic 2011 (CSL'11).
Editor: Marc Bezem; pp. 20–34

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Plan of the Paper. In section §2 we introduce and motivate the concept of transfinite update procedure and give a very short non-constructive proof of the existence of finite zeros.

In section §3 we formulate the notion of "learning process generated by an update procedure" and prove that every learning process terminates with a zero for the associated update procedure. The result represents a semi-constructive proof of the existence of finite zeros and the learning processes generated are "optimal", in the sense that one may provide constructively the best possible ordinal bounds to their length and to the size of finite zeros (by applying techniques of Mints et al. [8]).

In sections §4 and §5 we formalize the notion of update procedure in typed lambda calculus plus bar recursion and prove the existence of zeros for update procedures of ordinal less than ω^ω by writing down simple bar recursive terms. These results are enough to give computational interpretation to proofs of Elementary Analysis and Ramified Analysis, as formulated in [8], [9] (see the full version of this paper [1]). In fact, our methods yield bar recursive proofs of termination for epsilon substitution method as formulated in [9].

Acknowledgments. I'd like to thank Paulo Oliva for his advice throughout this work.

2 Learning in Predicative Analysis

It is very well-known and of fundamental importance that intuitionistic Arithmetic is constructive. This in particular implies that from a proof that there exists an object with a given property, one can always extract a computer program that construct an object with that property. Such feature of intuitionistic Arithmetic depends on the fact that all its axioms and inference rules never assert the existence of something that has not already been implicitly constructed.

In the classical framework, the situation is much different. From the computational point of view, any classical predicative subsystem of second order Arithmetic poses very difficult problems. The axioms that it adds on top of intuitionistic Arithmetic are ontologically very strong. For every formula $\phi(x)$, there is an axiom of comprehension asserting the existence of a function g able to decide the truth of $\phi(x)$:

$$\exists g^{\mathtt{N}\to\mathtt{N}}\forall x^{\mathtt{N}}.\ g(x)=0 \leftrightarrow \phi(x)$$

Axioms of countable choice assert the existence of functions computing existential witnesses of truth of formulas:

$$(\forall x^{\mathtt{N}}\exists y^{A}\ \phi(x,y)) \to \exists g^{\mathtt{N}\to A}\forall x^{\mathtt{N}}\phi(x,g(x))$$

In order to give a *natural* computational interpretation even for the most simple form of the excluded middle

$$\mathsf{EM}_1 : \forall n^{\mathtt{N}}.\exists x^{\mathtt{N}} Pnx \vee \forall y^{\mathtt{N}}\neg Pny$$

one would have to provide a program for deciding, given any number n, the truth of the formula $\exists x^{\mathtt{N}} Pnx$, with P decidable.

Given the situation, a direct computational interpretation of classical logic might seem impossible. Fortunately, there is a fundamental observation that enables us to partially solve this problem: whatever the function or the decision procedure whose existence is assumed, it will be used only a finite number of times in actual computations of finite results! In other words, non-computable functions exist – and we cannot do anything about that – but one only needs to compute finite approximations of them in order to carry out actual computations.

Over this observation, what we call "learning-based computational interpretations" of classical Arithmetic build their success. We include in this category Hilbert's epsilon substitution method, Avigad's update procedures [4] and Aschieri and Berardi's learning-based realizability [3] for intuitionistic Arithmetic plus EM_1.

2.1 Learning Based Realizability for Intuitionistic Arithmetic with EM_1

In the case of learning-based realizability for Intuitionistic Arithmetic with EM_1, one just considers a class of realizers recursive in the oracle for the Halting problem. Such programs easily interpret EM_1, but are ineffective; to recover effectiveness they are evaluated with respect to finite oracle approximations. Since approximations may be inadequate, results of computations may be wrong. But learning-based realizers are *self-correcting* programs, able to identify wrong oracle values used during computations and to correct them with right values that they learn during the same computations. Realizers keep correcting mistakes until they find a good finite approximation of the oracle they use.

2.2 Avigad's Finite Update Procedures for Peano Arithmetic

If one wants to provide a direct computational interpretation of excluded middle EM for arbitrary arithmetical formulas

$$\forall n^{\mathbb{N}}.\ \exists x_1^{\mathbb{N}} \forall y_1^{\mathbb{N}} \dots \exists x_k^{\mathbb{N}} \forall y_k^{\mathbb{N}}\ P(n, x_1, y_1, \dots, x_k, y_k) \vee \forall x_1^{\mathbb{N}} \exists y_1^{\mathbb{N}} \dots \forall x_k^{\mathbb{N}} \exists y_k^{\mathbb{N}} \neg P(n, x_1, y_1, \dots, x_k, y_k)$$

he needs much more computational power: an oracle for the Halting problem is no longer enough. For example, if one wants to interpret

$$\mathsf{EM}_2 := \forall n^{\mathbb{N}}.\ \exists x^{\mathbb{N}} \forall y^{\mathbb{N}} P(n, x, y)\ \vee\ \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} \neg P(n, x, y)$$

he also needs an oracle for the Halting problem for programs *recursive* in the oracle for the Halting problem. This is due to the fact that in order to check, for any fixed m, whether the formula $\forall y^{\mathbb{N}} P(n, m, y)$ is true, one can use a program $p(m)$ recursive in the oracle of the Halting problem. But in order to determine the truth of $\exists x^{\mathbb{N}} \forall y^{\mathbb{N}} P(n, x, y)$, one has to determine whether there exists an m such that $p(m)$ outputs the answer that the formula $\forall y^{\mathbb{N}} P(n, m, y)$ is true.
In general, in order to interpret EM_n one needs a sequence of oracles $\Phi_0, \dots, \Phi_n$ such that for every $1 \leq i \leq n$, Φ_i is an oracle for the Halting problem for programs *recursive* in the subsequence $\{\Phi_j\}_{1 \leq j < i}$. More precisely, let $T_i(x, y, z)$ the predicate, recursive in $\{\Phi_j\}_{1 \leq j < i}$, that holds iff the x-th Kleene's partial recursive function f_x, recursive in $\{\Phi_j\}_{1 \leq j < i}$, terminates on input y after z reduction steps. Then Φ_i must satisfy the following Skolem axiom:

$$\forall x^{\mathbb{N}}, y^{\mathbb{N}}, z^{\mathbb{N}}.\ T_i(x, y, z) \to T_i(x, y, \Phi_i(|x, y|))$$

where $|\_, \_|$ is a bijective coding of pairs of natural numbers into natural numbers.

Now, an Avigad update procedure is a functional $\mathcal{U}$ that takes as argument a finite sequence of functions $f_0, \dots, f_n$ approximating some oracles $\Phi_0, \dots, \Phi_n$. Then, it uses those functions to compute some witnesses for some provable Σ_1^0 formula of Peano Arithmetic PA (i.e. a formula of the form $\exists x^{\mathbb{N}} P(x)$, with P computable). Afterwards, it checks whether the result of its computation is sound. If it is not, it identifies some wrong value of f_i used in the computation and it corrects it with a new one. $\mathcal{U}$, remarkably, can always do that, thanks to the fact that in this case an instance of the Skolem axiom for Φ_i, for some i, (computed by substituting its oracles with their approximation $f_0, \dots, f_n$) must be false. But if an instance

$$T_i(n, m, l) \to T_i(n, m, f_i(|n, m|))$$

is false, then its antecedent is true and its consequent is false. Therefore, $\mathcal{U}$ *learns* a new value for f_i on argument $|n, m|$, namely l, which will replace its former wrong value $f_i(|n, m|)$.

▶ **Definition 1** (Avigad's Finite Update Procedures)**.** A *k-ary update procedure*, $k \in \mathbb{N}^+$, is a continuous function $\mathcal{U} : (\mathbb{N} \to \mathbb{N})^k \to \mathbb{N}^3 \cup \{\emptyset\}$ (i.e., its output is determined by a finite number of values of the input functions) such that the following holds:

1. for all function sequences $f = f_1, \ldots, f_k$

$$\mathcal{U}f = (i, n, m) \implies 1 \leq i \leq k$$

2. for all function sequences $f = f_1, \ldots, f_k$ and $g = g_1, \ldots, g_k$, for all $1 \leq i < k$, if
   - $\mathcal{U}f = (i, n, m)$
   - for all $j < i$, $f_j = g_j$
   - $g_i(n) = m$

   then: $\mathcal{U}g = (i, h, l) \implies h \neq n$.

If $\mathcal{U}$ is a k-ary update procedure, a *zero* for $\mathcal{U}$ is a sequence $f = f_1, \ldots, f_k$ of functions such that $\mathcal{U}f = \emptyset$.

Condition (2) of definition 1 means that the values of the i-th function depend on the values of some of the functions f_j, with $j < i$, and learning on level i is possible only if all the lower levels j have "stabilized". In particular, if $\mathcal{U}$ is a k-ary update procedure and $f : (\mathbb{N} \to \mathbb{N})^k$ is a sequence of functions approximating the oracles $\Phi_1, \ldots, \Phi_k$, there are two possibilities: either f is a fine approximation and then $\mathcal{U}f = \emptyset$; or f is not and then $\mathcal{U}f = (i, n, m)$, for some numerals n, m: $\mathcal{U}$ says the function f_i should be updated as to output m on input n. Moreover, if $\mathcal{U}f = (i, n, m)$, one in a sense has *learned* that $\Phi_i(n) = m$: by definition of update procedure, if g is a function sequence agreeing with f in its first $i - 1$ elements, g_i is another candidate approximation of Φ_i and $g_i(n) = m$, then $\mathcal{U}g$ does not represent a request to modify the value of g_i at point n, for $\mathcal{U}g = (i, h, l)$ implies $h \neq n$.

The main theorem about update procedures is that they always have zeros and these latter can be computed through learning processes *guided* by the former. Intuitively a zero of an update procedure represents a good approximation of the oracles used in a computation, and in particular a good enough one to yield some sought classical witness.

2.3 Transfinite Update Procedures for Predicative Systems of Analysis

In general, learning based computational interpretations of predicative fragments of classical analysis (see Mints et al. [8], [9]) provide answers to the computational challenges of classical axioms by the following three-stage pattern:

1. They identify a sequence $\{\Phi_\beta\}_{\beta<\alpha}$ – possibly transfinite – of non-computable functions $\mathbb{N} \to \mathbb{N}$.
2. They define classical witnesses for provable Σ^0_1 formulas by using programs recursive in Φ.
3. They define update procedures through which it is possible to find, for every particular computation, a suitable finite approximation of the functions of Φ such that one can effectively compute the witnesses defined at stage two.

The functions in the sequence Φ of stage (1) are the computational engine of the interpretation. Given the difficulty of computing witnesses in classical Arithmetic, they are always non-computable. It is therefore not surprising that given this additional computational

power, one is able to define at stage (2) witnesses for classical formulas. If we picture the sequence Φ as a sequence of infinite stacks of numbers, the learning process of point (3) finds a "vertical" approximation of Φ: functions of Φ are infinite stacks of numbers whereas their finite approximations are finite stacks. Moreover, a crucial point is that the sequence Φ is not an arbitrary sequence. In a sense, Φ is also "horizontally" approximated: for every ordinal α, the recursion theoretic Turing degree of Φ_α is approximated by the degrees of Φ_β, for $\beta < \alpha$. This property is very important: in this way, the values of the functions in Φ can be gradually approximated and learned.

More precisely, Φ can be seen as a sequence of functions obtained by transfinite iteration of recursion theoretic *jump* operator (see for example Odifreddi [10]). That is, for every β, if β is a successor, Φ_β has the same Turing degree of an oracle for the halting problem for the class of functions recursive in $\Phi_{\beta-1}$ (jump); if β is limit, Φ_β has the same Turing degre of the function mapping the code of a pair (α, n), with $\alpha < \beta$, into $\Phi_\alpha(n)$ (join or β-jump). A fundamental property of such a sequence is that the assertion $\Phi_\beta(n) = m$ depends only on the values of the functions Φ_α, for $\alpha < \beta$, and the values of Φ_β are learnable in the limit[1] by a program g recursive in the join of Φ_α for $\alpha < \beta$, which is a guarantee that the learning processes will terminate.

We now give an informal example of the kind of analysis which is needed to carry out the first stage of a learning-based interpretation, in the case of Elementary Analysis EA.

▶ **Example 2** (Elementary Analysis)**.** Consider a subsystem EA of second order Peano Arithmetic in which second order quantification is intended to range over arithmetical sets and hence over arithmetical formulas (formulas with only numerical quantifiers and possibly set parameters). Since one has to interpret excluded middle over arbitrary formulas, it is necessary to provide *at least* programs that can decide truth of formulas. Numerical quantifiers correspond to Turing jumps. That is, if we have a program t (with the same function parameters of ϕ) such that for every pair of naturals n, m

$$t(n, m) = \texttt{True} \iff \phi(n, m)$$

then the truth of $\exists x^{\mathbb{N}} \phi(n, x)$ is equivalent to the termination of a program $Q(n)$ exhaustively checking

$$t(n, 0), t(n, 1), t(n, 2), \ldots$$

until it finds - if there exists - an m such that $t(n, m) = \texttt{True}$. Applying the jump operator to the Turing degree t belongs to, one can write down a program χ_t which is able to determine whether $Q(n)$ terminates. That is

$$\chi_t(n) = \texttt{True} \iff \exists x^{\mathbb{N}} \phi(n, x)$$

Similarly, one eliminates universal numerical quantifiers, thanks to the fact that $\forall \equiv \neg\exists\neg$. Iterating these reasoning and applying $2k$ times the jump operator - and given a recursive enumeration $\phi_0, \phi_1, \ldots$, of arithmetical formulas - one can obtain for every Σ^0_{2k} formula

$$\phi_n(m) := \exists x_1^{\mathbb{N}} \forall y_1^{\mathbb{N}} \ldots \exists x_k^{\mathbb{N}} \forall y_k^{\mathbb{N}}\ P(m, x_1, y_1, \ldots, x_k, y_k)$$

a program t_n such that

$$t_n(m) = \texttt{True} \iff \phi_n(m)$$

[1] In the sense of Gold [7]: $\Phi_\beta(n) = m \iff \lim_{k\to\infty} g(n, k) = m$. We remark that we need here the sequence of oracles $\{\Phi_\alpha\}_{\alpha<\beta}$; without it, it would not be to possible learn values of the powerful Φ_β.

Using the ω-jump operator, one can write down a program u such that

$$u(n,m) = \texttt{True} \iff t_n(m) = \texttt{True}$$

and hence

$$u(n,m) = \texttt{True} \iff \phi_n(m)$$

Now a Σ^1_1 formula $\exists f^{\mathtt{N}\to\mathtt{Bool}}\phi_i$ - provided we assume that $f^{\mathtt{N}\to\mathtt{Bool}}$ ranges over arithmetical predicates - can also be expressed as

$$\exists n^{\mathtt{N}} t_i[\lambda m^{\mathtt{N}} u(n,m)/f]$$

So applying again a jump operator to the recursive degree of $t_i[\lambda m^{\mathtt{N}} u(n,m)/f]$, one is able to write a program determining the truth value of $\exists f^{\mathtt{N}\to\mathtt{Bool}}\phi_i$. Iterating this reasoning, one can decide the truth of arbitrary Σ^1_n formulas.
Summing up, in order to decide truth in Elementary Analysis, one needs to apply the jump operator $\omega + \omega$ times and thus produces a sequence Φ of non-computable functions Φ of length $\omega + \omega$. All the programs that we have described can be thought as recursive in some initial segment of Φ.

We are now in a position to introduce our axiomatization of the learning procedures cited in point (3) above.

▶ **Definition 3** (Transfinite Update Procedures)**.** Let $\alpha \geq 1$ be a numerable ordinal. An *update procedure of ordinal* α is a function $\mathcal{U} : (\alpha \to (\mathbb{N} \to \mathbb{N})) \to (\alpha \times \mathbb{N} \times \mathbb{N}) \cup \{\emptyset\}$ such that:

1. $\mathcal{U}$ is continuous. i.e. for any $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ there is a finite subset A of $\alpha \times \mathbb{N}$ such that for every $g : \alpha \to (\mathbb{N} \to \mathbb{N})$ if $f_\gamma(n) = g_\gamma(n)$ for every $(\gamma, n) \in A$, then $\mathcal{U}f = \mathcal{U}g$.

2. For all functions $f, g : \alpha \to (\mathbb{N} \to \mathbb{N})$ and every ordinal $\beta \in \alpha$, if
   - $t(f) = (\beta, n, m)$
   - for all $\gamma < \beta$, $f_\gamma = g_\gamma$
   - $g_\beta(n) = m$

   then: $t(g) = (\beta, i, j) \implies i \neq n$

The concept of transfinite update procedure is a generalization of Avigad's notion of finite update procedure. A transfinite update procedure, instead of taking just a finite number of function arguments, may get as input an arbitrary transfinite sequence of functions, which are intended to approximate a target sequence Φ; as output, it may return an update (β, n, m), which means that the β-th function taken as argument is an inadequate approximation of Φ_β and must be updated as to output m on input n. Condition (2) means that the values for the β-th function depend only on the values of functions of ordinal less than β in the input sequence and an update procedure returns only updates which are *relatively verified* and hence need not to be changed. In this sense, if $\mathcal{U}f = (\beta, n, m)$, one has *learned* that $\Phi_\beta(n) = m$; so if g_β is a candidate approximation of Φ_β and $g_\beta(n) = m$, then $\mathcal{U}g$ does not represent a request to modify the value of g_β at point n, whenever f and g agree on all ordinals less than β.
We remark that the choice of the type for an update procedure is somewhat arbitrary: we could have chosen it to be

$$(\alpha \to (X \to Y)) \to (\alpha \times X \times Y) \cup \{\emptyset\}$$

as long as the elements of the sets X and Y can be coded by finite objects. Since such coding may always be performed by using natural numbers, we choose to consider $X = Y = \mathbb{N}$.

The use of transfinite update procedures made by learning-based computational interpretations of classical arithmetic can be described as follows. Suppose those interpretations are given a provable formula with an attainable computational meaning, for example one of the form $\forall x^{\mathbb{N}} \exists y^{\mathbb{N}}\ Pxy$, with P decidable. Then, for every numeral n, they manage to define a term $t_n : (\alpha \to (\mathbb{N} \to \mathbb{N})) \to \mathbb{N}$ and an update procedure $\mathcal{U}_n$ of ordinal α such that

$$\mathcal{U}_n(f) = \emptyset \implies Pn(t_n(f))$$

for all $f : \alpha \to (\mathbb{N} \to \mathbb{N})$. The idea is that a witness for the formula $\exists y^{\mathbb{N}} Pny$ is calculated by t_n with respect to a particular approximation f of the oracle sequence Φ we have previously described. If the formula $Pn(t_n(f))$ is true, there is nothing to be done. If it is false, then $\mathcal{U}_n(f) = (\beta, n, m)$ for some β, n, m: a new value for Φ_β is learned. This is what we call "learning by counterexamples": from every failure a new positive fact is acquired. An instance of this kind of learning appears in the case on learning-based realizability for $\mathsf{HA} + \mathsf{EM}_1$, when one defines realizability for atomic formulas: in that case the pair (n, m) is produced by the realizer of the excluded middle. We have seen another example in the section on Avigad update procedures for PA and will see a further one in the full version of this paper ([1]) in the case $\alpha = \omega + k$, with $k \in \mathbb{N}$: the triple (β, n, m) will be produced through the evaluation of Skolem axioms for epsilon terms in the system EA. In general, the Skolem axioms used to define oracles are those who make possible learning by counterexamples.

The effectiveness of the above approach depends on the fact that every update procedure has a finite zero, as defined below.

▶ **Definition 4** (Finite Functions, Finite Zeros, Truncation and Concatenation of Function Sequences)**.** Let $\mathcal{U}$ be an update procedure of ordinal α.

1. $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ is said to be a *finite function* if the set of (γ, n) such that $f_\gamma n \neq 0$ is finite.
2. A *finite zero* for $\mathcal{U}$ is a finite function $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ such that $\mathcal{U}f = \emptyset$.
3. Let $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ and $\beta < \alpha$. Let $f_{<\beta} : \beta \to (\mathbb{N} \to \mathbb{N})$ be the *truncation* of f at β:
$$f_{<\beta} := \gamma \in \beta \mapsto f_\gamma$$
4. Let α_1, α_2 be two ordinals, $f_1 : \alpha_1 \to (\mathbb{N} \to \mathbb{N})$ and $f_2 : \alpha_2 \to (\mathbb{N} \to \mathbb{N})$. Then the *concatenation* $f_1 * f_2 : (\alpha_1 + \alpha_2) \to (\mathbb{N} \to \mathbb{N})$ of f and g is defined as:
$$(f * g)_\gamma(n) := \begin{cases} f_\gamma(n) & \text{if } \gamma < \alpha_1 \\ g_\beta(n) & \text{if } \gamma = \alpha_1 + \beta < \alpha_1 + \alpha_2 \end{cases}$$
5. With a slight abuse of notation, a function $f : \mathbb{N} \to \mathbb{N}$ will be sometimes identified with the corresponding length-one sequence of functions $0 \mapsto (n \in \mathbb{N} \mapsto f(n))$.

We now prove that every update procedure has a finite zero. We will give other more and more constructive proofs of this theorem, that will allow to compute finite zeros for update procedures and thus witnesses for classically provable formulas, thanks to learning-based interpretations. But for now we are only interested in understanding the reason of the theorem's *truth* and give a very short non-constructive proof. All the subsequent proofs can be seen as more and more sophisticated and refined constructivizations of the following argument.

▶ **Theorem 5** (Zero Theorem for Update Procedures of Ordinal α). *Let $\mathcal{U}$ be an update procedure of ordinal α. Then $\mathcal{U}$ has a finite zero.*

Proof. We define, by transfinite induction, a function $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ as follows. Suppose we have defined $f_\gamma : \mathbb{N} \to \mathbb{N}$, for every $\gamma < \beta$. Define the sequence $f_{<\beta} : \beta \to (\mathbb{N} \to \mathbb{N})$ of them all

$$f_{<\beta} := \gamma \in \beta \mapsto f_\gamma$$

Then define

$$f_\beta(x) = \begin{cases} 0 & \text{if } \forall g^{(\alpha-\beta)\to(\mathbb{N}\to\mathbb{N})}\ \forall z^{\mathbb{N}}\ \mathcal{U}(f_{<\beta} * g) \neq \langle \beta, x, z\rangle \\ y & \text{otherwise, for some } y \text{ such that } \exists g^{(\alpha-\beta)\to(\mathbb{N}\to\mathbb{N})}\ \mathcal{U}(f_{<\beta} * g) = \langle \beta, x, y\rangle \end{cases}$$

By axiom of choice and classical logic, for every β, $f_{<\beta}$ and f_β are well defined. So we can let

$$f := f_{<\alpha}$$

Suppose $\mathcal{U}(f) = \langle \beta, x, z\rangle$, for some $\beta < \alpha$: we show that it is impossible. For some $h : (\alpha - (\beta + 1)) \to (\mathbb{N} \to \mathbb{N})$, $f = f_{<\beta} * f_\beta * h$. Hence, for some $g : (\alpha - \beta) \to (\mathbb{N} \to \mathbb{N})$

$$\mathcal{U}(f_{<\beta} * g) = \langle \beta, x, y\rangle \wedge f_\beta(x) = y$$

by definition of f. But $\mathcal{U}$ is an update procedure and so

$$(\mathcal{U}(f_{<\beta} * g) = \langle \beta, x, y\rangle \wedge f_\beta(x) = y \wedge \mathcal{U}(f_{<\beta} * f_\beta * h) = \langle \beta, x, z\rangle) \implies x \neq x$$

which is impossible. We conclude that $\mathcal{U}(f) = \emptyset$ and, by continuity, that $\mathcal{U}$ has a finite zero. ◀

3 Learning Processes Generated by Transfinite Update Procedures

In this section we show that every update procedure $\mathcal{U}$ generates a learning process and this learning process always terminates with a finite zero of $\mathcal{U}$. This result is an abstract version of the termination of the H-process as defined in the various versions of epsilon substitution method (see Mints et al. [8]). The proof of termination is semi-constructive and is similar to the one in Mints et al. [8] (which however is by contradiction while ours is not).

If $\mathcal{U}$ is an update procedure and $\mathcal{U}(f) = \langle \gamma, n, m\rangle$, then the value of f_γ at argument n must be updated as to be equal to m. But as explained in the introduction, we may imagine that all the values of all the functions f_β, with $\beta > \gamma$, depend on the values of the *current* f_γ. Therefore, if we change some of the values of f_γ, we must erase all the values of all the functions f_β, for $\beta > \gamma$, because they may be inconsistent with the new values of f_β. In a sense, f is a fragile structure, that may be likened to an *house of cards*: if we change something in a layer, then all the higher ones collapse. We define an update operator $\oplus$ that performs those operations.

▶ **Definition 6** (Controlled Update of Functions). Let $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ and $\langle \gamma, n, m\rangle \in \alpha \times \mathbb{N} \times \mathbb{N}$. We define a function $f \oplus \langle \gamma, n, m\rangle : \alpha \to (\mathbb{N} \to \mathbb{N})$ such that

$$(f \oplus \langle \gamma, n, m\rangle)_\beta(x) := \begin{cases} f_\beta(x) & \text{if } \beta < \gamma \text{ or } (\beta = \gamma \text{ and } x \neq n) \\ m & \text{if } \gamma = \beta \text{ and } x = n \\ 0 & \text{otherwise} \end{cases}$$

We also define $f \oplus \emptyset := f$.

We now define the concept of "learning process generated by an update procedure $\mathcal{U}$". It may be thought as a process of updating and learning new values of functions, which is *guided* by $\mathcal{U}$. It corresponds to the step three of the learning based computational interpretations of classical arithmetic we have described in the introduction. Intuitively, such a learning process starts from the always zero function 0^α. If $\mathcal{U}$ says that some value of 0^α must be updated - i.e. $\mathcal{U}(0^\alpha) = \langle \gamma, n, m\rangle$ - then the learning process generates the function $\mathcal{U}^{(1)} := 0^\alpha \oplus \langle \gamma, n, m\rangle$. Similarly, if $\mathcal{U}$ says that some value of $\mathcal{U}^{(1)}$ must be updated - i.e. $\mathcal{U}(\mathcal{U}^{(1)}) = \langle \gamma', n', m'\rangle$ - then the learning process generates the function $\mathcal{U}^{(2)} := \mathcal{U}^{(1)} \oplus \langle \gamma', n', m'\rangle$. The process goes on indefinitely in the same fashion.

▶ **Definition 7** (Learning Processes Generated by $\mathcal{U}$)**.** Let $\mathcal{U}$ be an update procedure of ordinal α. For every $n \in \mathbb{N}$, we define a function $\mathcal{U}^{(n)} : \alpha \to (\mathbb{N} \to \mathbb{N})$ by induction as follows:

$$\mathcal{U}^{(0)} := 0^\alpha := \gamma \in \alpha \mapsto (n \in \mathbb{N} \mapsto 0)$$
$$\mathcal{U}^{(n+1)} := \mathcal{U}^{(n)} \oplus \mathcal{U}(\mathcal{U}^{(n)})$$

Moreover, a function $f : \alpha \to (\mathbb{N} \to \mathbb{N})$ is said to be $\mathcal{U}$-*generated* if there exists an n such that $f = \mathcal{U}^{(n)}$.

The aim of the learning process generated by $\mathcal{U}$ is to find a finite zero for $\mathcal{U}$. Indeed, if for some n, $\mathcal{U}(\mathcal{U}^{(n)}) = \emptyset$, then for all $m \geq n$, $\mathcal{U}^{(m)} = \mathcal{U}^{(n)}$ and we thus say that the learning process *terminates*. We now devote ourselves to the proof that learning processes always terminate. In other words, every update procedure $\mathcal{U}$ has a $\mathcal{U}$-generated finite zero.

Given an update procedure $\mathcal{U}$, its useful to define a new "simpler" update procedure, obtained from $\mathcal{U}$ by fixing some initial segment of its input, ignoring all updates relative to this fixed part of the input and adjusting their indices.

▶ **Definition 8.** Let $\mathcal{U}$ be an update procedure of ordinal α. Then, for any function $g : \beta \to (\mathbb{N} \to \mathbb{N})$, with $\beta < \alpha$, define a function

$$\mathcal{U}_g : ((\alpha - \beta) \to (\mathbb{N} \to \mathbb{N})) \to (\alpha - \beta) \times \mathbb{N} \times \mathbb{N} \cup \{\emptyset\}$$

as follows:

$$\mathcal{U}_g(f) = \begin{cases} \langle \gamma, n, m\rangle & \text{if } \mathcal{U}(g * f) = \langle \beta + \gamma, n, m\rangle \\ \emptyset & \text{otherwise} \end{cases}$$

(We point out that if $\beta = 0 = \emptyset$, $\mathcal{U}_g = \mathcal{U}$ as it should be)

Indeed $\mathcal{U}_g$ as defined above is an update procedure.

▶ **Lemma 9.** *Let $\mathcal{U}$ be an update procedure of ordinal α. Then, for any function $g : \beta \to (\mathbb{N} \to \mathbb{N})$, with $\beta < \alpha$:*

1. *$\mathcal{U}_g$ is an update procedure of ordinal $\alpha - \beta$.*
2. *For every $h : \mathbb{N} \to \mathbb{N}$, $\mathcal{U}_{g*h} = (\mathcal{U}_g)_h$.*

Proof. Immediate. ◀

The strategy of our termination proof can be described as follows. Given an update procedure $\mathcal{U}$ of ordinal α, we shall define a sequence of functions $g : \alpha \to (\mathbb{N} \to \mathbb{N})$ such that a "reduction lemma" can be proved: if, for some $\beta < \alpha$, $\mathcal{U}_{g_{<\beta}}$ has a $\mathcal{U}_{g_{<\beta}}$-generated finite zero, then for some $\gamma < \beta$ also $\mathcal{U}_{g_{<\gamma}}$ has a $\mathcal{U}_{g_{<\gamma}}$-generated finite zero. But the greater the ordinal β the easier it is to compute with a learning process a finite zero for $\mathcal{U}_{g_{<\beta}}$, because

the sequence $g_{<\beta}$ becomes so long that the input for $\mathcal{U}_{<\beta}$ becomes short. So we shall be able to show that for some large enough β, $\beta < \alpha$, $\mathcal{U}_{g_{<\beta}}$ has a $\mathcal{U}_{g_{<\beta}}$-generated finite zero, which proves the theorem in combination with the reduction lemma. This technique can be seen as a generalization of Avigad's [4] to the transfinite case.

We now prove the reduction lemma in the limit case.

▶ **Lemma 10** (Reduction Lemma, Limit Case). *Let $\mathcal{U}$ be an update procedure of ordinal α and $g : \beta \to (\mathbb{N} \to \mathbb{N})$, with β limit ordinal and $\beta < \alpha$. Then*

1. *If $f : (\alpha - \beta) \to (\mathbb{N} \to \mathbb{N})$ is $\mathcal{U}_g$-generated, then there exists $\gamma < \beta$ such that $0^{\beta-\gamma} * f$ is $\mathcal{U}_{g_{<\gamma}}$-generated.*

2. *If $\mathcal{U}_g$ has a $\mathcal{U}_g$-generated finite zero, then there exists $\gamma < \beta$ such that $\mathcal{U}_{g_{<\gamma}}$ has a $\mathcal{U}_{g_{<\gamma}}$-generated finite zero.*

Proof. See the full version of this paper [1]. ◀

We now prove the reduction lemma in the successor case.

▶ **Lemma 11** (Reduction Lemma, Successor Case). *Let $\mathcal{U}$ be an update procedure of ordinal α. Define $g : \mathbb{N} \to \mathbb{N}$ as follows:*

$$g(x) := \begin{cases} y & \text{if } \exists i.\ \mathcal{U}(\mathcal{U}^{(i)}) = \langle 0, x, y\rangle \wedge i = \min\{n \mid \exists z\ \mathcal{U}(\mathcal{U}^{(n)}) = \langle 0, x, z\rangle\} \\ 0 & \text{otherwise} \end{cases}$$

Then:

1. *For every finite function $g_0 \leq {}^2 g$, if $g_0 * 0^{\alpha-1}$ is $\mathcal{U}$-generated and f is $\mathcal{U}_{g_0}$-generated, then $g_0 * f$ is $\mathcal{U}$-generated.*

2. *If $\mathcal{U}_g$ has a $\mathcal{U}_g$-generated finite zero, then $\mathcal{U}$ has a $\mathcal{U}$-generated finite zero.*

Proof. See the full version of this paper [1]. ◀

We are now able to prove the main theorem: update procedures generate terminating learning processes.

▶ **Theorem 12** (Termination of Learning Processes). *Let $\mathcal{U}$ be an update procedure of ordinal α. Then, $\mathcal{U}$ has a finite zero. In particular, there exists $k \in \mathbb{N}$ such that $\mathcal{U}(\mathcal{U}^{(k)}) = \emptyset$.*

Proof. See the full version of this paper [1]. ◀

4 Spector's System B and Typed Update Procedures of Ordinal ω^k

Zeros of trasfinite update procedures cannot in general be computed in Gödel's system T: as we will show, already update procedures of ordinal $\omega + k$, with $k \in \omega$, can be used to give computational interpretation to Elementary Analysis and hence their zeros can be used to compute the functions provably total in Elementary Analysis. We will show however that Spector's system B is enough to compute zeros.

[2] We define $g_0 \leq g$ iff for all x $g_0(x) \neq 0 \implies g_0(x) = g(x)$

▶ **Definition 13** (Bar Recursion Operator, Spector's System B, Type Level of Bar Recursion)**.** In the following, we will work with Spector's system B (see Spector [12] and, for a modern exposition, Kohlenbach [6]) which is Gödel's T augmented with constants $\mathsf{BR}_{\tau,\sigma}, \Psi_{\tau,\sigma}$ respectively of type

$$T_1 \to T_2 \to T_3 \to T_4 \to \tau \qquad \text{and} \qquad T_1 \to T_2 \to T_3 \to T_4 \to \mathtt{Bool} \to \tau$$

with

$$T_1 = (\mathtt{N} \to \sigma) \to \mathtt{N}$$

$$T_2 = \sigma^* \to \tau$$

$$T_3 = \sigma^* \to (\sigma \to \tau) \to \tau$$

$$T_4 = \sigma^*$$

where σ^* is a type representing finite sequences of objects of type σ. The meaning of $\mathsf{BR}_{\tau,\sigma}$ is defined by the equation

$$\mathsf{BR}_{\tau,\sigma} YGHs \stackrel{\tau}{=} \begin{cases} Gs & \text{if } Y\hat{s} < |s| \\ Hs(\lambda x^{\sigma} \mathsf{BR}_{\tau,\sigma} YGH(s * x)) & \text{otherwise} \end{cases} \tag{1}$$

where $s * x$ denotes the finite sequence s followed by x, $\hat{s}$ denotes the function mapping n to s_n, if $n < |s|$, to 0^{σ} otherwise, where s_n is the n-th element of s and $|s|$ is the number of elements in s. If σ, τ, Y, G, H are determined by the context, we we will just write $\mathsf{BR}(s)$ in place of $\mathsf{BR}_{\tau,\sigma} YGHs$.
$\mathsf{BR}_{\tau,\sigma}$ is said to be *bar recursion of type* σ. The *type level* of bar recursion $\mathsf{BR}_{\tau,\mathtt{N}}$ of type $\mathtt{N}$ (said also type 0), is the type level of the constant $\mathsf{BR}_{\tau,\mathtt{N}}$, that is, assuming $\mathtt{N}^* = \mathtt{N}$, $\mathsf{max}(1, \mathsf{typelevel}(\tau)) + 2$.
In order to obtain a strongly normalizing system such that equation 1 holds, we have to add to system B the following reduction rules (for a proof of strong normalization, see Berger [5]):

$$\mathsf{BR}_{\tau,\sigma} YGHs \mapsto \Psi_{\tau,\sigma} YGHs(Y\hat{s} < |s|)$$
$$\Psi_{\tau,\sigma} YGHs(\mathtt{True}) \mapsto Gs$$
$$\Psi_{\tau,\sigma} YGHs(\mathtt{False}) \mapsto Hs(\lambda x^{\sigma} \mathsf{BR}_{\tau,\sigma} YGH(s * x))$$

where $<$ is a term coding the correspondent relation on natural numbers.

Since we are interested only in computable update procedures, we now fix a system for representing them. For the aim of computationally interpreting Elementary Analysis, update procedures can be assumed to belong to system T. However, for more powerful systems one may need more capable update procedures, so we define them to belong to B. Here, we limit ourselves to the ordinal ω^k, for $k \in \omega$, since this ordinal is enough to interpret Elementary Analysis and even fragments of Ramified Analysis (see for example, Mints et al. [9])

▶ **Definition 14** (Representation of Ordinals and Typed Update Procedures of Ordinal ω^k)**.** We will represent ordinal numbers of the form ω^k, with $k \in \omega$, by exploiting the order isomorphism between ω^k and $\mathbb{N}^k$ lexicographically ordered. So, for $k \in \omega$, $k > 0$, we set

$$[\omega^0] := \nu, [\omega^k] := \mathtt{N}^k$$

where ν is the empty string and

$$[\omega^0 \to (\mathbb{N} \to \mathbb{N})] := \mathtt{N} \to \mathtt{N}$$

and, if $k \in \omega$

$$[\omega^{k+1} \to (\mathbb{N} \to \mathbb{N})] := \mathtt{N} \to [\omega^k \to (\mathbb{N} \to \mathbb{N})]$$

where $\mathtt{N}$ is the type representing $\mathbb{N}$ in typed lambda calculus. Define moreover

$$[(\omega^k \times \mathbb{N} \times \mathbb{N}) \cup \{\emptyset\}] := [\omega^k] \times \mathtt{N} \times \mathtt{N}$$

Unfortunately, $\emptyset$ does not have a code. So we have to use an injective coding $|\_|$ of the set $(\omega^k \times \mathbb{N} \times \mathbb{N}) \cup \{\emptyset\}$ into the set of closed normal terms of type $[(\omega^k \times \mathbb{N} \times \mathbb{N}) \cup \{\emptyset\}]$. To fix ideas, we define $|(\beta, n, m)| = \langle \beta', n+1, m+1 \rangle$, with $\beta' : \mathtt{N}^k$ the code of β, and $|\emptyset| = \langle \langle 0, \ldots, 0 \rangle, 0, 0 \rangle$. A *typed update procedure of ordinal* ω^k is a term of Spector's system B of type:

$$[\omega^k \to (\mathbb{N} \to \mathbb{N})] \to [(\omega^k \times \mathbb{N} \times \mathbb{N}) \cup \{\emptyset\}]$$

satisfying point (2) of definition 3, where for simplicity function quantification is assumed to range over functions definable in system B. Equality as it appears in the definition is supposed to be extensional.

5 Bar Recursion Proof of the Zero Theorem for Typed Update Procedures of Ordinal ω^k

In this section we give a constructive proof of the Zero theorem for typed update procedures of ordinal less than ω^k. In particular we show that finite zeros can be computed with bar recursion of type 1. We start with the base case.

▶ **Theorem 15** (Zero Theorem for Update Procedures of Ordinal $1{=}\omega^0$)**.** *Let $\mathcal{U}$ be a typed update procedure of ordinal 1. Then $\mathcal{U}$ has a finite zero σ. Moreover, σ can be calculated as the normal form of a bar recursive term $\mathsf{Zero}(\mathcal{U})$ (defined uniformly on the parameter $\mathcal{U}$) of system T plus bar recursion of type 0.*

The result follows by Oliva [11]. We give below another proof, which is a simplification of Oliva's one, made possible by the slightly stronger condition we have imposed on the notion of update procedure.
The informal idea of the construction - but with some missing justifications - is the following. We reason over the well-founded tree of finite sequences of numbers s such that $\mathcal{U}(\hat{s}) = |(n, m)|$ and $n \geq |s|$. We want to construct a function $\sigma : \mathtt{N} \to \mathtt{N}$ which is a zero of $\mathcal{U}$. Suppose that we have constructed a "good" initial approximation $\sigma(0) * \cdots * \sigma(i)$ of σ; we want to prove that it can be extended to a long enough approximation of σ. Our first step is to continue with $\sigma(0) * \cdots * \sigma(i) * 0$. If this is a good guess, by well-founded induction hypothesis, we can extend $\sigma(0) * \cdots * \sigma(i) * 0$ to a complete approximation $\sigma(0) * \cdots * \sigma(n)$ of σ, with $n > i$. Since we are not sure that our previous guess was lucky, we compute $\mathcal{U}(\sigma(0) * \cdots * \sigma(n))$. If for all m

$$\mathcal{U}(\sigma(0) * \cdots * \sigma(n)) \neq |(i+1, m)|$$

then our approximation for $\sigma(i+1)$ is adequate, and we claim that $\sigma(0) * \cdots * \sigma(n)$ is the approximation of σ we were seeking. Otherwise

$$\mathcal{U}(\sigma(0) * \cdots * \sigma(n)) = |(i+1, m)|$$

for some m: $\mathcal{U}$ tells us that our guess for the value of $\sigma(i+1)$ is wrong. But now we know that $\sigma(0) * \cdots * \sigma(i) * m$ is a good initial approximation of σ and we have made progress. Again by well-founded induction hypothesis, we conclude that we can extend $\sigma(0) * \cdots * \sigma(i) * m$ to a good approximation of σ.

Proof. *of Theorem* 15. We formalize and complete the previous informal argument. In the following s will be a variable for finite sequences of numbers. Using bar recursion of type 0, we can define a term which builds directly the finite zero we are looking for and is such that:

$$\mathsf{BR}(s) = \begin{cases} \hat{s} & \text{if } \mathcal{U}\hat{s} = |(n,m)| \text{ and } n < |s| \\ \hat{s} & \text{if } \mathcal{U}\hat{s} = |\emptyset| \\ \mathsf{BR}(s * m) & \text{if } \mathcal{U}(\mathsf{BR}(s * 0)) = |(|s|, m)| \\ \mathsf{BR}(s * 0) & \text{if } \mathcal{U}(\mathsf{BR}(s * 0)) \neq |(|s|, m)| \text{ for all } m \end{cases}$$

(we assume that $\mathsf{BR}(s)$ checks in order every condition in its definition and executes the action corresponding to the first statisfied condition). We let σ be the normal form of

$$\mathsf{Zero}(\mathcal{U}) := \mathsf{BR}(\langle\rangle)$$

where $\langle\rangle$ is the empty sequence. Let us prove that σ is a finite zero of $\mathcal{U}$. Suppose $\mathcal{U}\sigma = |(n,m)|$: by showing that this is impossible, we obtain that $\mathcal{U}\sigma = |\emptyset|$. The normalization of $\mathsf{BR}(\langle\rangle)$ leads to the following chain of equations:

$$\begin{aligned} \mathsf{BR}(\langle\rangle) &= \mathsf{BR}(\sigma(0)) \\ &= \mathsf{BR}(\sigma(0) * \sigma(1)) \\ &\dots \\ &\dots \\ &= \mathsf{BR}(\sigma(0) * \dots * \sigma(i)) \\ &= \sigma(0)\widehat{\ *\cdots*\ }\sigma(i) \\ &= \sigma \end{aligned}$$

with

$$n < |\sigma(0) * \dots * \sigma(i)| = i + 1$$

In particular $\mathsf{BR}(\langle\rangle) = \mathsf{BR}(\sigma(0) * \dots * \sigma(n-1))$. Now, we have two cases:

1. $\mathcal{U}(\mathsf{BR}(\sigma(0) * \dots * \sigma(n-1) * 0)) = |(n,l)|$. Then

   $$\mathsf{BR}(\langle\rangle) = \mathsf{BR}(\sigma(0) * \dots * \sigma(n-1) * l)$$

   and so $\sigma(n) = l$, which is impossible, by definition 3 of update procedure, point (2), for $\mathcal{U}\sigma = |(n,m)|$.

2. for all l, $\mathcal{U}(\mathsf{BR}(\sigma(0) * \dots * \sigma(n-1) * 0)) \neq |(n,l)|$. Then by definition

   $$\mathsf{BR}(\sigma(0) * \dots * \sigma(n-1)) = \mathsf{BR}(\sigma(0) * \dots * \sigma(n-1) * 0)$$

   Therefore

   $$|(n,m)| = \mathcal{U}\sigma = \mathcal{U}(\mathsf{BR}(\langle\rangle)) = \mathcal{U}(\mathsf{BR}(\sigma(0) * \dots * \sigma(n-1) * 0))$$

   again impossible, by assumption of this case.

We have then proved that σ is the sought finite zero. ◀

We now prove that every typed update procedure of ordinal ω has a finite zero.

▶ **Theorem 16** (Zero Theorem for Typed Update Procedures of Ordinal ω)**.** *Let $\mathcal{U}$ be a typed update procedure of ordinal ω. Then $\mathcal{U}$ has a finite zero σ. Moreover, σ can be calculated as the normal form of a bar recursive term $\mathsf{Zero}_\omega(\mathcal{U})$ (defined uniformly on the parameter $\mathcal{U}$) of system T plus bar recursion of type $1 := \mathtt{N} \to \mathtt{N}$.*

Proof. The finite function $\sigma : [\omega \to (\mathbb{N} \to \mathbb{N})]$ we are going to construct can be represented as a finite function sequence $\sigma(0) * \sigma(1) * \cdots * \sigma(n)$, for a large enough n. In the following s is a variable ranging over finite sequences of natural number functions. Using bar recursion of type 1, we can define in a most simple way a term which builds directly the finite zero we are looking for. We present the construction gradually. To begin with, suppose we are able to define - uniformly on s - terms $\mathsf{BR}(s)$ and $g_s : (\mathtt{N} \to \mathtt{N})$ satisfying the following equation for every s:

$$\mathsf{BR}(s) = \begin{cases} \hat{s} & \text{if } \mathcal{U}\hat{s} = |(\gamma, n, m)| \text{ and } \gamma < |s| \\ \hat{s} & \text{if } \mathcal{U}\hat{s} = |\emptyset| \\ \mathsf{BR}(s * g_s) & \text{otherwise, where } \forall n, m \;\; \mathcal{U}(\mathsf{BR}(s * g_s)) \neq (|s|, n, m) \end{cases}$$

Let

$$\sigma := \mathsf{Zero}_\omega(\mathcal{U}) := \mathsf{BR}(\langle\rangle)$$

We prove that σ is a finite zero of $\mathcal{U}$. We show this by proving that $\mathcal{U}\sigma = (\gamma, n, m)$ is impossible. As in the proof of theorem 15

$$\mathsf{BR}(\langle\rangle) = \mathsf{BR}(\sigma(0) * \cdots * \sigma(i)) = \sigma(0) \widehat{* \cdots *} \sigma(i)$$

with $\gamma < i + 1$. Let

$$r := \sigma(0) * \cdots * \sigma(\gamma - 1)$$

By some computation

$$\begin{aligned} \mathcal{U}\sigma &= \mathcal{U}(\mathsf{BR}(\langle\rangle)) \\ &= \mathcal{U}(\mathsf{BR}(\sigma(0) * \cdots * \sigma(\gamma - 1))) \\ &= \mathcal{U}(\mathsf{BR}(r)) \\ &= \mathcal{U}(\mathsf{BR}(r * g_r)) \end{aligned}$$

Since by construction for all n, m

$$\mathcal{U}(\mathsf{BR}(r * g_r)) \neq |(|r|, n, m)| = |(\gamma, n, m)|$$

we obtain that $\mathcal{U}\sigma \neq (\gamma, n, m)$: impossible.
It remains to show that a g_s such that appears in the definition of $\mathsf{BR}(s)$ exists. Indeed, it is enough to set

$$g_s := \mathsf{Zero}(\lambda f^{\mathtt{N}\to\mathtt{N}} \mathcal{U}_{|s|}(\mathsf{BR}(s * f)))$$

where, for $i \in \mathbb{N}$, we have defined

$$\mathcal{U}_i := \lambda f^{\mathtt{N}\to(\mathtt{N}\to\mathtt{N})}. \text{ if } \mathcal{U}(f) = |(i, n, m)| \text{ then } |(n, m)| \text{ else } |\emptyset|$$

We prove now that in fact $\mathcal{U}(\mathsf{BR}(s * g_s)) \neq |(|s|, n, m)|$ for all n, m. First, observe again that for every s

$$\mathsf{BR}(s) = s * h_1 \widehat{* \cdots *} h_n$$

for some terms $h_1, \ldots, h_n$ of type $\mathtt{N} \to \mathtt{N}$. Now, fix any finite sequence s of type-$\mathtt{N} \to \mathtt{N}$ terms. We want to show that

$$F_s := \lambda f^{\mathtt{N}\to\mathtt{N}} \mathcal{U}_{|s|}(\mathsf{BR}(s * f))$$

is an update procedure of ordinal 1. Suppose $F_s g_1 = |(n, m)|$, $g_2(n) = m$ and $F_s g_2 = |(h, l)|$. Then, by definition of F_s, it must be that

$$\mathcal{U}(\mathsf{BR}(s * g_1)) = |(|s|, n, m)| \qquad \text{and} \qquad \mathcal{U}(\mathsf{BR}(s * g_2)) = |(|s|, h, l)|$$

Moreover,

$$\mathsf{BR}(s * g_2)_{|s|}(n) = g_2(n) = m$$

Since $\mathcal{U}$ is an update procedure, $h \neq n$ must hold; therefore F_s is an update procedure of ordinal 1. But by definition of g_s, Zero and theorem 15, this means that

$$|\emptyset| = F_s(\mathsf{Zero}(F_s)) = \mathcal{U}_{|s|}(\mathsf{BR}(s * g_s))$$

By definition of $\mathcal{U}_{|s|}$ it must be true that $\mathcal{U}(\mathsf{BR}(s * g_s)) \neq |(|s|, n, m)|$ for all n, m. ◀

The previous argument can be generalized in order to prove the Zero theorem for typed update procedures of ordinal ω^k.

▶ **Theorem 17** (Zero Theorem for Typed Update Procedures of Ordinal ω^k, with $k \in \omega$)**.** *Let $\mathcal{U}$ be a typed update procedure of ordinal ω^k. Then $\mathcal{U}$ has a finite zero σ. Moreover, σ can be calculated as the normal form of a bar recursive term $\mathsf{Zero}_{\omega^k}(\mathcal{U})$ (defined uniformly on the parameter $\mathcal{U}$) of system T plus bar recursion of some type A, where $\mathsf{typelevel}(A) = 1$.*

Proof. The idea is to break the zero we want to construct in ω blocks of length ω^{k-1} and build each block by using Zero_{ω^k}. See the full version of this paper [1]. ◀

References

1 F. Aschieri, *Learning in Predicative Analysis*, chapter 6 of [2]

2 F. Aschieri, *Learning, Realizability and Games in Classical Arithmetic*, PhD Thesis, 2011 http://arxiv.org/abs/1012.4992

3 F. Aschieri, S. Berardi, *Interactive Learning-Based Realizability for Heyting Arithmetic with* EM_1, Logical Methods in Computer Science, 2010

4 J. Avigad, *Update Procedures and 1-Consistency of Arithmetic*, Mathematical Logic Quarterly, vol. 48, 2002

5 U. Berger, *Continuous Semantics for Strong Normalization*, LNCS 3526, pag. 23-34, 2005

6 U. Kohlenbach, *Applied Proof Theory*, Springer-Verlag, Berlin, Heidelberg, 2008

7 E. M. Gold, *Limiting Recursion*, Journal of Symbolic Logic 30, pag. 28-48, 1965

8 G. Mints, S. Tupailo, W. Bucholz, *Epsilon Substitution Method for Elementary Analysis*, Archive for Mathematical Logic, volume 35, 1996

9 G. Mints, S. Tupailo, *Epsilon Substitution Method for the Ramified Language and* Δ^1_1*-Comprehension Rule*, Logic and Foundations of Mathematics, 1999

10 P. Odifreddi, *Classical Recursion Theory*, Studies in Logic and Foundations of Mathematics, Elsevier, 1989

11 P. Oliva, *Understanding and Using Spector's Bar Recursive Interpretation of Classical Analysis*, Proceedings of CiE'2006, LNCS, vol. 3988, Springer, 2006

12 C. Spector, *Provably Recursive Functionals of Analysis: a Consistency Proof of Analysis by an Extension of Principles in Current Intuitionistic Mathematics*, Dekker (ed.), Recursive Function Theory: Proceedings of Symposia in Pure Mathematics, vol. 5. AMS, Providence, 1962