Queen Mary, University of London
School of Electronic Engineering and Computer Science

LEARNING, REALIZABILITY AND GAMES IN CLASSICAL ARITHMETIC

Federico Aschieri

PhD Thesis

Supervisors: prof. Stefano Berardi and dr. Paulo Oliva

Date: 25 March, 2011

ABSTRACT. In this dissertation we provide mathematical evidence that the concept of *learning* can be used to give a new and intuitive computational semantics of classical proofs in various fragments of Predicative Arithmetic.

First, we extend Kreisel modified realizability to a classical fragment of first order Arithmetic, Heyting Arithmetic plus EM_1 (Excluded middle axiom restricted to Σ^0_1 formulas). We introduce a new realizability semantics we call "Interactive Learning-Based Realizability". Our realizers are *self-correcting* programs, which learn from their errors and evolve through time, thanks to their ability of perpetually questioning, testing and extending their knowledge. Remarkably, that capability is entirely due to classical principles when they are applied on top of intuitionistic logic.

Secondly, we extend the class of learning based realizers to a classical version $\mathcal{PCF}_{\text{Class}}$ of $\mathcal{PCF}$ and, then, compare the resulting notion of realizability with Coquand game semantics and prove a full soundness and completeness result. In particular, we show there is a one-to-one correspondence between realizers and recursive winning strategies in the 1-Backtracking version of Tarski games.

Third, we provide a complete and fully detailed constructive analysis of learning as it arises in learning based realizability for $\mathsf{HA}+\mathsf{EM}_1$, Avigad's update procedures and epsilon substitution method for Peano Arithmetic PA. We present new constructive techniques to bound the length of learning processes and we apply them to reprove - by means of our theory - the classic result of Gödel that provably total functions of PA can be represented in Gödel's system T.

Last, we give an axiomatization of the kind of learning that is needed to computationally interpret Predicative classical second order Arithmetic. Our work is an extension of Avigad's and generalizes the concept of update procedure to the transfinite case. Transfinite update procedures have to learn values of transfinite sequences of non computable functions in order to extract witnesses from classical proofs.

A Margherita

.

“Alors tous deux on est repartis
Dans le tourbillon de la vie
On à continué à tourner
Tous les deux enlacés
Tous les deux enlacés
Tous les deux enlacés.”

Acknowledgments

This dissertation would simply not exist without the contribution of my supervisors, Stefano Berardi and Paulo Oliva.

Stefano, I have to thank you so much. You have deeply influenced my work with your profound ideas on proof theory: you know, my accomplishment would not have been possible if I hadn't started from your great insights. You also have invested an extraordinary amount of time in checking my work, teaching me how to write mathematics, supporting and helping me in many situations, well beyond your duties.

Paulo, thanks for your support, your brilliant guidance and suggestions: you have led me to exciting research in my year at Queen Mary.

I wish to thank also Gianluigi Bellin: without you too this thesis would not have been produced. You have supported and helped me greatly in the early stages of my studies. You have introduced me to my future topics of research and had a determinant influence on the direction of my career. Thanks also for our dinners in London.

Ringrazio anche la mia famiglia, per la presenza costante, per tutto l'affetto e il supporto che non mi hanno mai fatto mancare.

Infine, grazie Margherita, per essere speciale come sei, per avermi voluto bene sempre, perché quando si tratta di te niente passa, niente scorre, niente cambia: questa tesi è dedicata a te, come il mio cuore lo è da quando ci siamo conosciuti.

Contents

CHAPTER 1

Introduction

1.1. A Computational Semantics of Classical Proofs

In this dissertation we provide mathematical evidence that the concept of *learning* can be used to give a new and intuitive computational semantics of classical proofs in various fragments of Predicative Arithmetic. The main definite result in this sense is a new realizability semantics for $\mathsf{HA} + \mathsf{EM}_1$, which we call "Interactive Learning Based Realizability" (shortly, *learning based realizability*). $\mathsf{HA} + \mathsf{EM}_1$ is first order intuitionistic Heyting Arithmetic with the principle of excluded middle over Σ^0_1 formulas, a classical axiom that eluded computational semantics for a long time and, probably, never had an intuitive one.

EM_1 has already been extensively studied in terms of Curry-Howard correspondence (see [43]), that is, in terms of computational constructs that can be associated to it in order to extract constructive information from classical proofs. Thanks to this approach, the constructive content of classical logic can be interestingly attained in terms of *proof transformations*, or *reduction rules* applied to the corresponding computational constructs. While these results are satisfying from the computational point of view, we feel they are not satisfying in terms of human *understanding*. Though this issue may at first appear marginal, it is not: classical proofs can now be "executed" thanks to Curry-Howard correspondence, but extracted programs are still very difficult to understand. Nowadays, there are research programs whose aim is precisely to *understand* programs extracted from classical proofs, since without this understanding programs cannot be analyzed, neither optimized nor improved, to begin with. Without high level grasp of a program, it is like having a black box, that in some mysterious way gives always correct answers. This phenomenon probably happens because classical principles are well understood only in terms of the computational devices they are associated to. Therefore, when one has to interpret a classical proof, he is forced to formalize it and then to extract the corresponding program. While this approach is feasible with small proofs, it is unmanageable with complex ones without great technical effort. This seems to explain why computational interpretations of classical logic are still not universally used by mathematicians and computer scientists.

As for ourselves, we think that proof theory should offer a *proof semantics*: that is, not only a way of extracting the computational content of classical proofs, but *also* a high level explanation of what are the general ideas used by the programs extracted from proofs and what is the constructive *meaning* of quantifiers, logical connectives and axioms in the framework of classical logic. Only in this way, the general mathematician or computer scientist could extract intuitively the computational content hidden in some proof he wishes to analyze. Such a semantics was put forward a long time ago for intuitionistic logic - starting from Heyting semantics, passing through Kleene realizability to arrive to Kreisel modified realizability - but we think that in the case of classical logic there are still no ultimate results.

In this dissertation, we put a major effort to lay the ground for a *proof semantics* of Predicative Arithmetic based on the concept of *learning*. A learning based realizability interpretation of $\mathsf{HA}+\mathsf{EM}_1$ will be completely explained and formalized, compared with game semantics and thoroughly analyzed by constructive means. We will define a computational model of "intelligent" *self-correcting* programs, which learn from their errors and evolve through time, thanks to their ability of perpetually questioning, testing and extending their knowledge. Remarkably, that capability is entirely due to classical principles when they are applied on top of intuitionistic logic. We shall thus conclude that the computational content of classical fragment $\mathsf{HA}+\mathsf{EM}_1$ can be described in terms of learning. Moreover, we will introduce a more general concept of learning *by levels*, that generalizes Avigad's one [5] and will serve as a foundation for possible extensions of our learning based realizability to full first order Peano Arithmetic and even Predicative Analysis. The learning based computation interpretation of classical logic is a new and exciting field of research, but just started: in this thesis we give substantial contributions, but the path to follow is still long.

In this chapter we give an overview of what are proof transformations, what are proofs semantics and what are the general ideas of learning in classical logic. We start with a short review of known results about intuitionistic logic, in order to explain how the issue "proof transformation vs. proof semantics" was solved in that case. We then explain the ideas behind epsilon substitution method, Coquand's game semantics and Avigad's update procedures, the three major sources of inspiration for our work and where the concept of learning first appeared, implicitly in the first case, explicitly and much more elegantly in the second and the third. These pioneering contributions are also the underpinnings of the more recent work of Berardi and de' Liguoro [11], which will be the starting point of this thesis. We conclude with a synopsis of the contribution and the structure of our dissertation.

1.2. Proof Transformations and Proof Semantics in Intuitionistic Arithmetic

Intuitionistic logic was the main method of reasoning used in mathematics until the eighteenth century. Existence of objects with special properties was mainly established by explicitly constructing objects with those properties. With Dedekind, Cantor, Hilbert, Brouwer, Frege and many others, non constructive reasoning became central, thanks to its great conceptual and simplifying power. Non constructive methods were immediately questioned by mathematicians such as Kronecker and others, but soon became generally employed and accepted. After the famous days of paradoxes in set theory and mathematics, however, a new season of concerns and foundational efforts began in mathematical logic. Brouwer, in particular, advocated the need for *intuitionistic logic*, which rejected some classical principles such as the excluded middle and impredicative definitions.

Intuitionistic logic was born as a *constructive logic*, but what does that mean? Syntactically, it is presented just as a set of axioms and inference rules in a formal language. And the reason why, for example, excluded middle is left out remains inevitably obscure, if a semantics for assertions in intuitionistic logic is not provided; the excluded middle is a very natural and intuitive principle, after all, known in logic since Aristotle.

The issue is informally solved through the so called *Heyting semantics* (see for example Girard [21]). According to Brouwer, in order to assert a statement, one has to provide some sort of *construction*. What is a construction? The following is Heyting's answer:

(1) A construction of an atomic formula is a *computation* showing its truth.

(2) A construction of $A \wedge B$ is a pair formed by a construction of A and a construction of B.

(3) A construction of $A \vee B$ is a pair whose first element is a boolean i such that: if $i = \texttt{True}$, then the second element of the pair is a construction of A, if $i = \texttt{False}$, then the second element of the pair is a construction of B.

(4) A construction of $A \to B$ is an algorithm taking as input a construction of A and returning as output a construction of B.

(5) A construction of $\forall x^{\mathbb{N}} A(x)$ is an algorithm taking as input a natural number n and returning as output a construction of $A(n)$.

(6) A construction of $\exists x^{\mathbb{N}} A(x)$ is a pair whose first element is a number n such that the the second element of the pair is a construction of $A(n)$.

Thanks to Heyting semantics, one immediately recognizes why the excluded middle $A \vee \neg A$ is rejected by intuitionistic logic: a construction of $A \vee \neg A$ would require to decide which one among $A, \neg A$ is true, which is not generally possible in algorithmic way. In Heyting semantics, each logical connective and quantifier is given a computational meaning, which is enough clear as to be intuitively used by humans with the aim of devising constructive proofs.

1.2.1. Proof Transformations in HA. Let us consider the formal system HA, Heyting Arithmetic, which is the intuitionistic version of the usual first order Peano Arithmetic PA. Since HA is born as a constructive logic, one expects it to be sound with respect to Heyting semantics. But since as a formal system HA does not mention at all the concept of Heyting construction, it is not obvious, given any provable formula, how to extract a construction from any of its proofs.

The first method for implicitly extracting such a computational content is due to Gentzen [19], [20] and falls under the category of *proof transformations*. The idea is the following. One defines a set of transformation rules mapping proofs of a formula into proofs of the same formula. He then iterates these rules from the initial proof until he finds a proof with a special syntactical structure, and from this proof he obtains the constructive content "implicit" in the initial proof. We will not consider Gentzen technique (called "cut elimination") but instead a later one due to Prawitz, known as *normalization*, which applies to natural deduction proofs. We start from recalling the inference rules of HA.

(1) $\dfrac{}{A}$ where A is a Peano axiom.

(2) $\dfrac{A \quad B}{A \wedge B}$ $\qquad \dfrac{A \wedge B}{A}$ $\qquad \dfrac{A \wedge B}{B}$

(3) $\dfrac{A \to B \quad A}{B}$ $\qquad \dfrac{B}{A \to B}$

(4) $$\frac{A}{A \vee B} \qquad \frac{B}{A \vee B}$$
$$\frac{A \vee B \quad C \quad C}{C}$$

(5) $$\frac{\forall \alpha^{\mathtt{N}} A}{A[t/\alpha]} \; \frac{A}{\forall \alpha^{\mathtt{N}} A}$$
where t is a term in the language of HA

(6) $$\frac{A[t/\alpha^{\mathtt{N}}]}{\exists \alpha^{\mathtt{N}}.A} \qquad \frac{\exists \alpha^{\mathtt{N}}.A \quad C}{C}$$

(7) $$\frac{A(0) \quad \forall \alpha^{\mathtt{N}}.A(\alpha) \to A(\mathsf{S}(\alpha))}{\forall \alpha^{\mathtt{N}} A}$$

A natural deduction proof is a tree whose nodes are formulas obtained from their children by inference rules. Leaves of a deduction tree are either hypotheses or discharged hypotheses and all the standard management of discharged ones is assumed here (see for example van Dalen [15]). As usual, quantifier rules must be applied under the usual restrictions on the quantified variables.

We are interested in a set of rules that allow to transform any proof of any formula A in a special *normal form* proof of A. We limit ourselves to the rules for implication and induction and we refer to [15] for a complete treatment. The proof transformations defined are the following. In any deduction, a sub-deduction

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_1\\ B\end{array}}{A \to B} \qquad \begin{array}{c}\mathcal{D}_2\\ A\end{array}}{B}$$

is replaced by

$$\begin{array}{c}\mathcal{D}_1[\mathcal{D}_2/A]\\ B\end{array}$$

where $\mathcal{D}_1[\mathcal{D}_2/A]$ is the deduction obtained from D_1 by replacing all the hypotheses A with the deduction $\mathcal{D}_2$ of A. Moreover, in any deduction, a sub-deduction

$$\dfrac{\begin{array}{c}\mathcal{D}_1\\ A(0)\end{array} \qquad \begin{array}{c}\mathcal{D}_2\\ \forall \alpha^{\mathtt{N}}.A(\alpha) \to A(\mathsf{S}(\alpha))\end{array}}{\dfrac{\forall \alpha^{\mathtt{N}} A}{A(n)}}$$

where n is a numeral, is replaced by a deduction $\mathcal{D}^n$ of $A(n)$, where by induction $\mathcal{D}^0$ is defined as

$$\begin{array}{c}\mathcal{D}_1\\ A(0)\end{array}$$

and for any numeral m, $\mathcal{D}^{\mathsf{S}(m)}$ is defined as

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_2\\ \forall\alpha^{\mathtt{N}}.A(\alpha)\to A(\mathsf{S}(\alpha))\end{array}}{A(m)\to A(\mathsf{S}(m))} \quad \begin{array}{c}\mathcal{D}^m\\ A(m)\end{array}}{A(\mathsf{S}(m))}$$

It is possible to show that any natural deduction proof without hypotheses of a closed formula A can be transformed into a closed normal proof of A. Just by inspection and some straigthforward reasoning on the resulting proof shape, one can show that if $A = \exists x^{\mathtt{N}} B$, then the last inference rule must be of the form

$$\dfrac{B[t/\alpha^{\mathtt{N}}]}{\exists\alpha^{\mathtt{N}}.B}$$

Hence one automatically finds a witness t for $\exists x^{\mathtt{N}} B$. If $A = B_1 \vee B_2$, then the last inference rule must be of the form

$$\dfrac{B_i}{B_0 \vee B_1}$$

with $i \in 0, 1$: again a witness a required by Heyting semantics. The constructive information of the original proof of A can thus be found by normalization.

In general, by normalization, it is possible to show that every formula provable in HA has a construction, in the sense of Heyting. There is a remark to be done: without Heyting semantics, the process of normalization would be perfectly non intelligible. One would only see a sequence of transformations performed on a initial proof until a normal form proof is found and, magically, a witness pops out of nothing. Moreover, normalization is not a technique that can be used in an intuitive way, because it requires full formalization of proofs and then to understand how the normalization process proceeds.

1.2.2. Proof Semantics for HA. We now explain how a formal proof semantics for HA can be formulated in terms of Kreisel modified realizability [34]. Modified realizability is a formalization of Heyting semantics, carefully carved and designed for HA^ω (but we shall consider its restriction to HA). The idea is to restrict the class of algorithms used in Heyting's definition: only algorithm representable in Gödel's system T (see chapter 2) are allowed. This is an important restriction: only bounded iteration is explicitly used by such algorithms and this property rules out the possibility that the computational content of proofs may be found by blind search and other constructively unjustified techniques (as it might happen with trivial Kleene-style realizers [31]).

We start by associating types to formulas as to mirror the structure of the programs used in Heyting semantics.

Definition 1.2.1 . (Types for realizers) For each arithmetical formula A we define a type $|A|$ of T by induction on A:

(1) $|P(t_1, \ldots, t_n)| = \mathtt{N}$,

(2) $|A \wedge B| = |A| \times |B|$,

(3) $|A \vee B| = \mathtt{Bool} \times (|A| \times |B|)$,

(4) $|A \to B| = |A| \to |B|$,

(5) $|\forall x A| = \mathtt{N} \to |A|$,

(6) $|\exists x A| = \mathtt{N} \times |A|$

We now define the Kreisel modified realizability relation $t \Vdash C$. It is clear that a modified realizer represents a construction in the sense of Heyting.

Definition 1.2.2 (Modified Realizability). Assume t is a closed term of Gödel's system T (see chapter 2), C is a closed formula, and $t : |C|$. Let $\vec{t} = t_1, \ldots, t_n : \mathtt{N}$. We define the relation $t \Vdash C$ by induction and by cases according to the form of C:

(1) $t \Vdash P(\vec{t})$ if and only if $P(\vec{t}) = \mathtt{True}$

(2) $t \Vdash A \wedge B$ if and only if $\pi_0 t \Vdash A$ and $\pi_1 t \Vdash B$

(3) $t \Vdash A \vee B$ if and only if either $p_0 t = \mathtt{True}$ and $p_1 t \Vdash A$, or $p_0 t = \mathtt{False}$ and $p_2 t \Vdash B$

(4) $t \Vdash\!\Vdash A \to B$ if and only if for all u, if $u \Vdash A$, then $tu \Vdash_s B$

(5) $t \Vdash \forall x A$ if and only if for all numerals n, $tn \Vdash A[n/x]$

(6) $t \Vdash \exists x A$ if and only for some numeral n, $\pi_0 t = n$ and $\pi_1 t \Vdash A[n/x]$

Thanks to Kreisel modified realizability and to the correspondence between typed lambda terms and natural deduction proofs, one can give a new intuitive meaning to the normalization process for intuitionistic proofs. One does that by decorating inference rules with terms of system T in such a way that each of the natural deduction transformation rules we have previously seen correspond to a normalization step in the associated term. We shall present in chapter 3 the full decoration, here we consider only the case for $\to$-rules:

$$\frac{u \vdash A \to B \quad t \vdash A}{ut \vdash B} \qquad \frac{u \vdash B}{\lambda x^{|A|} u \vdash A \to B}$$

We see that, finally, the idea of Heyting construction, in the form of modified realizability, is explicitly associated to the proof. Each inference rule either uses a realizer to compute something or defines a realizer. If we consider a decorated deduction $\mathcal{D}$

$$\dfrac{\dfrac{\begin{matrix}\mathcal{D}_1\\ u \vdash B\end{matrix}}{\lambda x^{|A|} u \vdash A \to B} \qquad \begin{matrix}\mathcal{D}_2\\ t \vdash A\end{matrix}}{(\lambda x^{|A|} u)t \vdash B}$$

the previously described proof transformation rule of $\mathcal{D}$ in $\mathcal{D}_1[\mathcal{D}_2/A]$ corresponds to the normalization step $(\lambda x^{|A|} u)t = u[t/x]$, which is simply the evaluation of a function at its argument.

The point to be made here is the following. First, one has a *local* interpretation of each inference rule in terms of realizability, that helps to understand what the rule itself means

and what it is going to accomplish from the computational point of view. Moreover, every axiom and every formula in general is given a computational meaning through realizability. In this way, one understands what arithmetical assertions mean from the constructive standpoint and what one has to do in order to constructively prove them. Without such a semantics, it would not be possible for a human to understand in a simple way what is the intuitive computational content that a proof offers. The goal of this dissertation is to extend this result to $\mathsf{HA} + \mathsf{EM}_1$.

1.3. Learning in Classical Arithmetic: A Brief History

In this dissertation we provide a realizability semantics of $\mathsf{HA} + \mathsf{EM}_1$ classical proofs in terms of learning. The first question we have to answer is therefore the following: *what is to be learned by realizers of classical proofs*? Surprisingly, the answer was anticipated a long time ago by Hilbert and his epsilon substitution method.

1.3.1. Learning in the Epsilon Substitution Method. Kreisel's modified realizers have not enough computational power for deciding truth of formulas and thus they are prevented to realize classical principles such as the excluded middle; this limitation in turn prevents them to decorate classical proofs with the aim of finding witnesses for classically provable existential statements. However, as proven by Kreisel himself [33], one can extract from a classical proof in PA of any Π^0_2 formula $\forall x^{\mathbb{N}} \exists y^{\mathbb{N}} Pxy$, with P decidable, a non trivial algorithm that given any number n finds a witness m such that Pnm is true. This result implies that even non constructive proofs of existence hide constructive information.

Hilbert's idea (for a modern account, see [5]) to circumvent the apparent inability of constructive methods to computationally interpret classical proofs, at least for Π^0_2 provable formulas, was to introduce non computable functions and to define through them classical witnesses. The role of those non computable functions, called epsilon substitutions, is to assign values to some non effectively evaluable *epsilon terms*. For each arithmetical formula $A(x)$ one introduces a term $\epsilon x A(x)$, whose intended denotation can be any number n such that $A(n)$ is true. Thus, $\epsilon x A(x)$ reads as "an x such that $A(x)$". For every term t, one has therefore an axiom

$$A(t) \to A(\epsilon x A(x))$$

which captures completely the intended meaning of an epsilon term. Such an axiom is said to be a *critical formula*. Epsilon terms are not effectively computable, so one starts with an epsilon substitution S that gives to them dummy values. In order to extract the computational content of a classical proof, one has to satisfy all the critical formulas appearing in the proof. Fortunately, there is only a finite number of them in any proof and so nothing in principle makes the goal impossible. It turns out that one can *learn* values of epsilon terms by counterexamples. Suppose, for instance, that a critical formula

$$A(t) \to A(\epsilon x A(x))$$

is false under some substitution S. Then, if we denote with $S(\epsilon x A(x))$ the value associated to $\epsilon x A(x)$ by S, we have that $A(S(\epsilon x A(x)))$ is false. However, if t evaluates to n under S, $A(n)$ must be true! So, one updates the substitution S as to give to $\epsilon x A(x)$ the value n, because he has *learned* a witness through a counterexample. Things, however, are not so easy: if there are many critical formulas, this attempt of making true one of them, may make false another. Hilbert's *Ansatz* (approach) was to show that a specially defined series of this learning steps must terminate.

Hilbert's idea is brilliant, but it has never been used to give a semantics of classical proofs. As to ourselves, we shall use it and we can now anticipate that the goal of our learning based realizers will be to *learn values of non computable functions*.

1.3.2. Learning in Coquand Game Semantics. The concept of learning in classical logic has been beautifully improved and reframed in terms of special Tarski games, in which the participating players are allowed to correct their moves: this is Coquand games semantics [14]. The interest, for our purposes, of this semantics, lies in the fact that the concept of recursive winning strategy in a standard Tarski game is nothing but a rephrasing of realizability: an adaptation of Tarski games to classical logic offers an opportunity to translate back the new intuitions in a realizability semantics, which is more suitable to proof interpretations.

We first review what is a *Tarski game* (actually, the concept was explicitly defined by Hintikka (see [29]) and it is a sort of folklore). In a Tarski game there are two players, Eloise and Abelard, and a negation-and-implication-free formula B on the board (we assume that for every atomic formula its negation can also be expressed as an atomic formula). Intuitively, Eloise tries to show that the formula is true, while Abelard tries to show that it is false. Turns, sets of possible moves and winners are defined accordingly to the form of B:

(1) If $B = \exists x^{\mathtt{N}} A(x)$, then Eloise has to choose a numeral n and B is replaced by the formula $A(n)$.

(2) If $B = \forall x^{\mathtt{N}} A(x)$, then Abelard has to choose a numeral n and B is replaced by the formula $A(n)$.

(3) If $B = A_1 \vee A_2$, then Eloise has to choose a numeral $i \in \{0, 1\}$ and B is replaced by the formula A_i.

(4) If $B = A_1 \wedge A_2$, then Abelard has to choose a numeral $i \in \{0, 1\}$ and B is replaced by the formula A_i.

(5) If B is atomic and true, Eloise wins the game, otherwise Abelard wins.

Informally, Eloise has a *recursive winning strategy* in this Tarski game, if she has an algorithm for choosing her next moves that enables her to win every play. It is possible to show that a formula has a construction in the sense of Heyting if and only if Eloise has a recursive winning strategy in its associated Tarski game. In this sense, Tarski games rephrase Heyting semantics.

It is not surprising, then, that Eloise does not have a recursive winning strategy for every instance of the excluded middle EM_1. Coquand solved this impasse by allowing Eloise to backtrack, i.e. to erase her moves and return to a previous state of the game (actually, when interpreting the cut rule, Coquand allowed also Abelard to backtrack, but we will not consider this case). Again, *learning* by counterexamples enters the scene. It is possible to prove that Eloise has a recursive winning strategy in the backtracking version of the Tarski game associated to EM_1. Suppose, for example, that

$$\mathsf{EM}_1 := \forall x^{\mathtt{N}}.\ \exists y^{\mathtt{N}} Pxy \vee \forall y^{\mathtt{N}} \neg Pxy$$

and consider any play: we show how Eloise can win. Abelard has to move and, for some n, chooses the formula

$$\exists y^{\mathbb{N}} Pny \vee \forall y^{\mathbb{N}} \neg Pny$$

Then it is the turn of Eloise, who believes that no witness for $\exists y^{\mathbb{N}} Pny$ can be found. So she chooses

$$\forall y^{\mathbb{N}} \neg Pny$$

Again, Abelard for some m chooses

$$\neg Pnm$$

If Pnm is false, then Eloise wins. But if Pnm is true, she would have lost the standard Tarski game. However, according to the new rules, she can now backtrack to a previous position. Observe that Abelard has falsified Eloise's belief in the non existence of witnesses for the formula $\exists y^{\mathbb{N}} Pny$, *actually providing* one witness. So Eloise backtracks to the position

$$\exists y^{\mathbb{N}} Pny \vee \forall y^{\mathbb{N}} \neg Pny$$

and this time chooses

$$\exists y^{\mathbb{N}} Pny$$

followed by

$$Pnm$$

and so she wins.

We remark how close is this kind of learning to the one in epsilon substitution method. In both cases, some formula wished to be true is false. But from its falsehood one can always learn a new positive fact: a witness for an existential statement.

1.3.3. Learning in Avigad's Update Procedures. In [5], Avigad has formulated an abstract axiomatization of learning as it implicitly appears in the epsilon substitution method for first order Peano Arithmetic. He has explicitly introduced the non computable functions needed by the epsilon method to elicit the computational content of classical proofs and formulated in a clear way the notion of *update procedure* that formalizes what we call "learning by levels".

We give a definition slightly different from Avigad's, but analogous. Intuitively, a k-ary update procedure, with $k \in \mathbb{N}^+$, is a functional which takes as input a finite sequence $f = f_1, \ldots, f_k$ of functions approximating some oracles $\Phi_1, \ldots, \Phi_k$, such that each one of those functions is defined in terms of the previous ones. Then, it uses those functions to compute some witnesses for some provable Σ^0_1 formula of PA. Afterwards, it checks whether the result of its computation is sound. If it is not, it identifies some wrong value $f_i(n)$ used in the computation and corrects it with a new one.

Definition 1.3.1 (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

i) for all $j < i$, $f_j = g_j$;

ii) $\mathcal{U}f = (i, n, m)$, $g_i(n) = m$ and $\mathcal{U}g = (i, h, l)$

then $h \neq n$.

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 1.3.1 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.

1.4. Learning in Classical Arithmetic: Contributions and Structure of This Dissertation

The aim of this dissertation is to study and *describe* the computational content of classical proofs in terms of *learning*. In particular, the contributions and the structure of this dissertation are the following.

1.4.1. Chapter 3. Our first contribution is to put together in a novel way the ideas contained in the epsilon substitution method, Coquand game semantics and Avigad's update procedures in order to define a realizability semantics of proofs which extends in a simple way Kreisel modified realizability to the classical system $\mathsf{HA} + \mathsf{EM}_1$: we shall call it (interactive) *learning based realizability*.

In a few words, learning based realizability describes a way of making oracle computations more effective, through the use of approximations of oracle values and learning of new values by counterexamples. A learning based realizer is in the first place a term of system $\mathcal{T}_{\text{Class}}$, which is a simple extension Gödel's system T plus some oracles of the same Turing degree of an oracle for the Halting problem. Of course, if a realizer was only this, it would be ineffective and hence useless. Therefore, learning based realizers are computed with respect to *approximations* of the oracles of $\mathcal{T}_{\text{Class}}$ and thus effectiveness is recovered. Since approximations may be sometimes inadequate, results of computations may be wrong. But a learning based realizer is also a *self-correcting* program, able to spot incorrect oracle values used during computations and to correct them with *right* values. The new values are *learned*, surprisingly, by realizers of EM_1 and all the oracle values needed during each particular computation are acquired through learning processes. Here is the fundamental insight: classical principle may be computationally interpreted as learning devices.

Our realizability semantics allows not only to extract realizers as usual by decorating classical proofs, but also to understand the intuitive meaning, behaviour and goals of the extracted realizers.

1.4.2. Chapter 4. Our second contribution is, first, to extend the class of learning based realizers to a classical version $\mathcal{PCF}_{\text{Class}}$ of $\mathcal{PCF}$ and, then, to compare the resulting notion of realizability with Coquand game semantics and prove a full soundness and completeness result. In particular, we show there is a one-to-one correspondence between realizers and recursive winning strategies in the 1-Backtracking version of Tarski games.

The soundness theorem should be useful to understand the significance and see possible uses of learning based realizability. The idea is that playing games represents a way of challenging realizers and of seeing how they react to the challenge by learning from failure and counterexamples.

The proof of the completeness theorem in our view, moreover, has an interesting feature. In a sense, it is the first application of the ideas of learning based realizability to a concrete non trivial classical proof, which is our version of the one given by Berardi et al. [9]. That proof classically shows that if Eloise has *recursive* winning strategy in the 1-Backtracking Tarski game associated to a formula A, then she also has a winning strategy in the Tarski game associated to A (but a non computable strategy, only recursive in an oracle for the Halting problem). We manage to associate a constructive content to this seemingly ineffective proof and find out that it hides a learning mechanism to gain correct oracle values from failures and counterexamples. We then transform this learning mechanism into a learning based realizer.

1.4.3. Chapter 5. Our third contribution is a complete and fully detailed constructive analysis of learning as it arises in learning based realizability for $\mathsf{HA}+\mathsf{EM}_1$, Avigad's update procedures and epsilon substitution method for Peano Arithmetic PA. We present new constructive techniques to bound the length of learning processes and we apply them to reprove - by means of our theory - the classic result of Kreisel that provably total functions of PA can be represented in Gödel's system T. An interesting novelty is that we develop type-theoretic techniques to reason and prove in a new way theorems about update procedures and epsilon substitution method. A notable byproduct of our work is the introduction of a "constructive" non standard model of Gödel's system T. Our analysis is also a first step toward an extension of learning based realizability to full PA.

1.4.4. Chapter 6. Our last contribution is an axiomatization of the kind of learning that is needed to computationally interpret Predicative classical second order Arithmetic. Our work is an extension of Avigad's and generalizes the concept of update procedure to the transfinite case. Transfinite update procedures have to learn values of transfinite sequences of non computable functions in order to extract witnesses from classical proofs. We shall present several proofs of the fact that transfinite update procedures have zeros. The last one uses methods of type theory and in particular bar recursion: the algorithms presented are powerful and yet quite simple. The interest of our results is twofold. First, we extend Avigad's intuitive description of the learning content of the epsilon substitution method to the second order case. Secondly, we take a first step toward the extension of learning based realizability to Predicative second order Arithmetic, since we have isolated the concept of learning that will have to be employed.

CHAPTER 2

Technical Preliminaries

2.1. Background

This dissertation is almost self-contained from the technical point of view. We cover the needed background here by reviewing what will be our main technical tool: Gödel's System T.

2.1.1. Gödel's system T. Gödel's system T (see [21], for example) is simply typed λ-calculus, enriched with natural numbers, booleans, conditional if_T and primitive recursion R_T in all types together with their associated reduction rules. We start by defining the types of system T.

Definition 2.1.1 (Types of System T). The set of *types* of system T is defined inductively as follows:

(1) $\mathtt{N}$ and $\mathtt{Bool}$ are types.

(2) If U, V are types, $U \times V$, $U \to V$ are types.

The type $\mathtt{N}$ represents the set $\mathbb{N}$ of natural numbers and $\mathtt{Bool}$ represents the set $\mathbb{B} = \{\mathrm{True}, \mathrm{False}\}$ of booleans, while product types $T \times U$ and arrows types $T \to U$ represent respectively cartesian products and function spaces. We assume $\to$ associates to the left:

$$T \to U \to V = T \to (U \to V)$$

Definition 2.1.2 (Terms of System T). We define the terms of system T inductively as follows:

(1) For all types U, the variables $x_0^U, \ldots, x_n^U, \ldots$ are terms of type U.

(2) 0 is a term of type $\mathtt{N}$. $\mathtt{True}$ and $\mathtt{False}$ are terms of type $\mathtt{Bool}$.

(3) For every type T, if_U is a term (constant) of type $U := \mathtt{Bool} \to T \to T \to T$. Terms of the form $\mathsf{if}_T t_1 t_2 t_3$ will be written in the more legible form $\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3$.

(4) For every type T, R_U is a term (recursion constant) of type $U := T \to (\mathtt{N} \to (T \to T)) \to \mathtt{N} \to T$. The type T in R_T will be omitted whenever inferable from the context.

(5) If t is of type $\mathtt{N}$, then $\mathsf{S}(t)$ is a term of type $\mathtt{N}$.

(6) If t and u are terms of types respectively $U \to V$ and U, then tu is a term of type V.

(7) If t is a term of type $U \times V$, then $\pi_0 t$ and $\pi_1 t$ are terms of types respectively U and V.

(8) If u and v are terms of types respectively U and V, then $\langle u, v\rangle$ is a term of type $U \times V$.

(9) If v is a term of type V and x^U a variable, then $\lambda x^U v$ is a term of type $U \to V$.

As usual, given two terms u and t and a variable x, with $u[t/x]$ we shall denote the term resulting from u by replacing all free occurrences of x in u with t, avoiding capture of variables.

We now give the reduction rules that explain the computational meaning of the syntax of system T.

Definition 2.1.3 (One Step β_0 Reduction and β Reduction). We define a binary relation β_0 between terms of system T as the least relation satisfying the following properties:

(1) If $u\ \beta_0\ u'$ and $v\ \beta_0\ v'$, then for all terms u, v, it holds that $uv\ \beta_0\ u'v$, $uv\ \beta_0\ uv'$, $\langle u, v\rangle\ \beta_0\ \langle u', v\rangle$ and $\langle u, v\rangle\ \beta_0\ \langle u, v'\rangle$.

(2) If $u\ \beta_0\ u'$, then $\lambda x^T u\ \beta_0\ \lambda x^T u'$ and $\pi_i u\ \beta_0\ \pi_i u'$ for $i = 0, 1$.

(3) $(\lambda x^T u)t\ \beta_0\ u[t/x]$.

(4) $\pi_i\langle u_0, u_1\rangle\ \beta_0\ u_i$ for $i = 0, 1$.

(5) $\mathsf{R}uv0\ \beta_0\ u$.

(6) $Ruv\mathsf{S}(t)\ \beta_0\ vt(\mathsf{R}uvt)$.

(7) $\mathsf{if}\ \mathtt{True}\ \mathsf{then}\ u_1\ \mathsf{else}\ u_2\ \beta_0\ u_1$ and $\mathsf{if}\ \mathtt{False}\ \mathsf{then}\ u_1\ \mathsf{else}\ u_2\ \beta_0\ u_2$.

(8) $t\ \beta\ t'$ if $t = t'$ or $t\ \beta_0\ u_1\ \beta_0\ u_2\ \beta_0 \ldots \beta_0\ u_n\ \beta_0\ t'$ for some terms $u_1, u_2 \ldots, u_n$.

(9) We say that t is in normal form if $t\ \beta_0\ t'$ does not hold, for every t'.

We now define the equality rules for terms of system T. Throughout the dissertation we will write $u = t$ if the equation is provable by means of the following rules.

Definition 2.1.4 (Equational Theory of $\mathcal{T}$). We list the axioms of equality for system T:

(1) $t = t$.

(2) If $t = u$, then $u = t$.

(3) If $t = u$ and $u = v$, then $t = v$.

(4) If $u = u'$ and $v = v'$, then $uv = u'v'$ and $\langle u, v\rangle = \langle u', v'\rangle$.

(5) If $u = u'$, then $\lambda x^V u = \lambda x^V u'$ and $\pi_i u = \pi_i u'$ for $i = 0, 1$.

(6) $(\lambda x^T u)t = u[t/x]$.

(7) $\pi_i\langle u_0, u_1\rangle = u_i$ for $i = 0, 1$.

(8) $\mathsf{R}uv0 = u$.

(9) $\mathsf{R}uv\mathsf{S}(t) = vt(\mathsf{R}uvt)$.

(10) $\mathsf{if}\ \mathtt{True}\ \mathsf{then}\ u_1\ \mathsf{else}\ u_2 = u_1$ and $\mathsf{if}\ \mathtt{False}\ \mathsf{then}\ u_1\ \mathsf{else}\ u_2 = u_2$.

Every term of Gödel's system T has a unique normal form (see [21]).

THEOREM 2.1.1 (NORMALIZATION AND CHURCH-ROSSER PROPERTY). *For every term t of system T there exists $n \in \mathbb{N}$ such that $t\ \beta_0\ t_1\ \beta_0 \dots \beta_0\ t_m$ implies $m \leq n$. Hence, t has a normal form.*
Moreover, if $t\ \beta\ t_1$ and $t\ \beta\ t_2$, then there exists t' such that $t_1\ \beta\ t'$ and $t_2\ \beta\ t'$. As consequence, t has a unique normal form.

A term is *closed* if it has no free variables; a *numeral* is a term of the form $\mathsf{S}^n(0)$, with $n \in \mathbb{N}$, having inductively defined $\mathsf{S}^0(0) := 0$ and $\mathsf{S}^{n+1}(0) := \mathsf{S}(\mathsf{S}^n(0))$. We will constantly use the following characterization of normal forms (see again [21]).

THEOREM 2.1.2 (NORMAL FORM CHARACTERIZATION FOR SYSTEM T). *Assume A is an atomic type. Then any closed normal term t of T of type A is either a numeral $n : \mathtt{N}$ or a boolean $\mathtt{True}, \mathtt{False} : \mathtt{Bool}$.*

CHAPTER 3

Interactive Learning-Based Realizability for Heyting Arithmetic with EM_1

ABSTRACT. In this chapter, we extend Kreisel modified realizability to a classical fragment of first order Arithmetic, Heyting Arithmetic plus EM_1 (Excluded middle axiom restricted to Σ^0_1 formulas). In particular, we introduce a new realizability semantics we call "Interactive Learning-Based Realizability". Our realizers are *self-correcting* programs, which are able to learn something from every failure and use the new knowledge to correct themselves. We build our semantics over Avigad's fixed point result [5], but the same semantics may be defined over different constructive interpretations of classical arithmetic (in [10], continuations are used). Our notion of realizability extends intuitionistic realizability and differs from it only in the atomic case: we interpret atomic realizers as "learning agents".

3.1. Introduction

From now on, we will call HA Heyting Intuitionistic Arithmetic, with a language including one symbol for each primitive recursive predicate or function (see [15] or section 3.3). We call Σ^0_1-formulas the set of all formulas $\exists x.P(x,y)$ for some primitive recursive predicate P, and EM_1 the *Excluded middle axiom restricted to* Σ^0_1*-formulas*. For a detailed study of the intuitionistic consequences of the sub-classical axiom EM_1 we refer to [1].

This chapter is based on Aschieri and Berardi [3]. We extend Berardi and de' Liguoro ([7], [10]) notion of atomic realizability - originally conceived for quantifier free primitive recursive Arithmetic plus EM_1 - to full predicate logic, namely Heyting Arithmetic with EM_1 ($\mathsf{HA} + \mathsf{EM}_1$). Our idea is to interpret classical proofs as constructive proofs on a suitable structure $\mathcal{N}$ for natural numbers and maps of Gödel's system T, by applying to the semantics of Arithmetic the idea of "finite approximation" used to interpret Herbrand's Theorem. We extend intuitionistic realizability to a new notion of realizability, which we call "Interactive learning-based Realizability". We provide a term assignment for the standard natural deduction system of $\mathsf{HA} + \mathsf{EM}_1$, which is surprisingly equal in all respects to that of HA, but for the fact that we have non-trivial realizers for atomic formulas and a new realizer for EM_1.

Our semantics is "local": we do not introduce a global variable representing an "external" goal, different for each particular proof one wants to interpret, as in continuation interpretation, in Friedman's A-translation and in Krivine's Classical Realizability. The goal of realizers is fixed, "internal", and is either to provide right constructions or to learn new information about excluded middle. In this way, we interpret EM_1 and thus classical proofs locally and step-by-step, in order to solve a major problem of all computational interpretations: global illegibility, which means that, even for simple classical proofs, it is extremely difficult to understand how each step of the extracted program is related to the ideas of the proof, and what it is the particular task performed by each subprogram of the

extracted program. The main sources of inspiration of this chapter are works of Kleene, Hilbert, Coquand, Hayashi, Berardi and de' Liguoro and Avigad.

Intuitionistic Realizability revisited. Recall chapter 1. In [31], Kleene introduced the notion of realizability, a formal semantics for intuitionistic arithmetic. Later, Kreisel [34] defined modified realizability, the same notion but with respect to Gödel's system T instead of Kleene's formalism of partial recursive functions. Realizability is nothing but a formal version of Heyting semantics for intuitionistic logic, translated into the language of arithmetic.

Intuitively, realizing a closed arithmetical formula A means exhibiting a computer program - called realizer - able to calculate all relevant information about the truth of A. Hence, realizing a formula $A \vee B$ means realizing A or realizing B, after calculating *which one* of the two is actually realized; realizing a formula $\exists x A(x)$ means computing a numeral n - called a witness - and realizing $A(n)$.

These two cases are indeed the only ones in which we have relevant information to calculate about the truth of the corresponding formula, and there is a decision to be made: realizing a formula $\forall x A$ means exhibiting an algorithm which takes as input a numeral n and gives as output realizers of $A(n)$; realizing a formula $A \wedge B$ means realizing A and realizing B; realizing $A \to B$ means providing an algorithm which takes as input realizers of A and gives realizers of B; in these cases we provide no information about the formula we realize and we only take the inputs we will use for realizing existential or disjunctive formulas. Finally, realizing an atomic formula means that the formula is true: in this case, the realizer does nothing at all.

Hence, intuitionistic realizability closely follows Tarski's definition of truth - the only difference being effectiveness: for instance, while Tarski, to assert that $\exists x A$ is true, contented himself to know that there exists some n such that $A(n)$ is true, Kleene asked for a program that calculates an n such that $A(n)$ is true.

Intuitionistic natural deduction rules are perfectly suited to preserve realizability. In order to actually build realizers from intuitionistic natural deductions, it suffices to give realizers for the axioms. Since our goal is to interpret classical connectives using Heyting and Kleene interpretation of intuitionistic connectives, then a first, quite naive idea would be the following: if we devised realizers for Excluded Middle, we would be able to extend realizability to all classical arithmetic.

Unfortunately, from the work of Turing it is well known that not every instance of Excluded Middle is realizable. If $Txyz$ is Kleene's predicate, realizing $\forall x \forall y . \exists z Txyz \vee \forall z \neg Txyz$ implies exhibiting an algorithm which for every n, m calculates whether or not the n-th Turing machine halts on input m: the halting problem would be decidable. Hence, there is no hope of computing with effective programs all the information about the truth of Excluded Middle.

However, not all is lost. A key observation is the following. Suppose we had a realizer O of the Excluded Middle and we made a natural deduction of a formula $\exists x A$ actually using Excluded Middle; then, we would be able to extract from the proof a program u, containing O as subprogram, able to compute the witness for $\exists x A$. Given the effectiveness of u, after a finite number of steps - and more importantly, after a finite number of calls to O - u would yield the required witness. It is thus clear that u, to perform the calculation, would use only a *finite piece of information about the Excluded Middle*. This fundamental fact gives us hope: maybe there is not always necessity of fully realizing Excluded Middle, since in finite computations only a finite amount of information is used. If we were able to gain

that finite information during the computation, as it is the case in the proof of Herbrand's Theorem, we could adapt intuitionistic realizability to Classical Logic.

Herbrand's Theorem and the idea of "finite approximation". (A corollary of) Herbrand's Theorem says that if a universal first order theory T, in a suitable language supporting definition by cases, proves a statement $\exists x P(x)$, then one can extract from any proof a term t and closed instances $A_1, \ldots, A_n$ of some universal formulas of T such that $A_1 \wedge \ldots \wedge A_n \rightarrow P(t)$ is a propositional tautology. So, even using classical logic, one can define witnesses. The problem is that the functions occurring in t may not be computable, because the language of T is allowed to contain arbitrary functions. However, given the finiteness of the information needed about any function used during any finite computation of t, in order to carry out actual calculations one would only have to find finite approximations of the non-computable functions involved, thus recovering effectiveness. We choose to follow this intuition: we will add non-computable functions to our language for realizers and exploit the existence of these ideal objects in order to find concrete computational solutions.

This general idea dates back to Hilbert's ϵ-substitution method (for a neat reformulation of the ϵ-method see for example Avigad [5]). As noted by Ackermann [2], the ϵ- substitution method may be used to compute witnesses of provable existential statements of first order Peano Arithmetic. The procedure is simple: introduce Skolem functions (equivalently, ϵ-terms) and correspondent quantifier free Skolem axioms in order to reduce any axiom to a quantifier free form; take a PA-proof of a sentence $\exists x P(x)$ and translate it into a proof using as axioms only universal formulas; then apply Herbrand's theorem to the resulting proof, obtaining a quantifier free proof of $P(t)$, for some term t of the extended language; finally, calculate a suitable finite approximation of the Skolem functions occurring in t and calculate from t an n such that $P(n)$ holds.

However, while proofs in quantifier free style are very simple combinatorial objects, they lose the intuitive appeal, the general concepts, the structure of high level proofs. Hence, it may be an impossible task to understand extracted programs. Moreover we have a computational syntactic method but no semantics of proofs and logical operators based on the idea of "finite approximation", as the realizability interpretations are based on the idea of "construction". However, in the ϵ-method, albeit only for quantifier free formulas, we see in action the method of intelligent *learning*, driven by the Skolem axioms used in the proofs. One of the aims of this chapter is to extend this "semantics of learning" from atomic propositions to individuals, maps, logical connectives and quantifiers of full natural deduction proofs. An important contribution comes from Coquand [14].

Coquand's Game Semantics for Classical Arithmetic. Computing all relevant information about the truth of a given formula A is not always possible. In [14] and in the context of game semantics, Coquand introduced a new key idea around this problem: the correspondence between backtracking (in game theory, retracting a move) and "learning", a refinement of the idea of "finite approximation". If we cannot compute all the right information about the truth of a formula, maybe we could do this if we were allowed to make finitely many mistakes and to learn from them.

Suppose, for instance, we have the formula $\forall x.\exists y Pxy \vee \forall y \neg Pxy$, but we have no algorithm which, for all numeral n given as input, outputs false if $\forall y \neg Pny$ holds and outputs true if $\exists y Pny$ holds. Then we may describe a learning algorithm r as follows. Initially, for all n given as input, r outputs false. Intuitively, r is initially persuaded - following the

principle "if I don't see, I do not believe" - that for all numeral n there is no numeral m such that Pnm holds. Hence, when asked for his opinion about the formula $\exists yPny \vee \forall y\neg Pny$, r always says: $\exists yPny$ is false. However, if someone - an opponent of r - to show that r is wrong, comes out with an m such that Pnm holds, r realizes indeed to be mistaken, and stores the information "Pnm is true". Then, the next time being asked for an opinion about $\exists yPny \vee \forall y\neg Pny$, r will say: true. In other words, such r, after at most one "mind changing", would be able to learn the correct answer to any question of the form: "which one among $\exists yPny$, $\forall y\neg Pny$ does hold?". This is actually learning by counterexamples and is the key idea behind Coquand's semantics.

Our question is now: can we formulate a realizability notion based on learning by counterexamples in order to extend Kreisel's interpretation to all individuals, maps and connectives of the sub-classical Arithmetic $\mathsf{HA} + \mathsf{EM}_1$? Following Hayashi [28], in our solution we modify the notion of individual, in such a way that individuals change with time, and realizers "interact" with them.

Hayashi's Proof Animation and Realizability. In [28], Hayashi explains a notion of realizability for a sub-classical arithmetic, called limit computable mathematics. Basing his analysis on ideas of Gold [22], he defines a Kleene's style notion of realizability equal to the original one but for the fact that the notion of individual changes: the witnesses of existential and disjunctive formulas are calculated by a stream of guesses and "learned in the limit" (in the sense that the limit of the stream is a correct witness). An individual a is therefore a computable map $a : \mathbb{N} \to \mathbb{N}$, with $a(t)$ representing the value of the individual at time t.

For instance, how would Hayashi realize the formula $\forall x.\exists yPxy \vee \forall y\neg Pxy$? He would define an algorithm H as follows. Given any numeral n, H would calculate the truth value of $\forall y \leq nPny$. Then the correct answer to the question: "which one among $\exists yPny$, $\forall y\neg Pny$ does hold?" is learned in the limit by computing $P(n,0)$, $P(n,1)$, $P(n,2)$,..., $P(n,k)$,... and thus producing a stream of guesses either of the form false, false, false,..., true, true,..., true,... or of the form false, false, false, ..., false, ..., the first stabilizing in the limit to true, the second to false. Hayashi's idea is to perform a completely blind and exhaustive search: in such a way, the correct answer is guaranteed to be eventually learned (classically). Hayashi's realizers do not learn in an efficient way: in Hayashi's notion of realizability the only learning device is to look through all possible cases. Instead, we want to combine the idea of individual as limit, taken from Hayashi, with notion of learning in which the stream of guesses is *driven by the proof itself*, as in Coquand's game semantics. For the quantifier-free fragment, this was done by Berardi [7] and Berardi-de' Liguoro [10].

Realizability Based on Learning: Berardi-de' Liguoro interpretation. We explain [10] using Popper's ideas [40] as a metaphor. According to Popper, a scientific theory relies on a set of unproved - and unprovable - hypotheses and, through logic, makes predictions susceptible of being falsified by experiments. If a prediction is falsified, some hypothesis is incorrect. In front of a counterexample to a theory's prediction, one must modify the set of hypotheses and build a better theory, which will be tested by experiments, and so on. Laws of Nature are universal statements, that cannot be verified, but are suitable to falsification. We may explain the link between falsifiable hypotheses and EM_1. For every n, given an instance $\exists y.Pny \vee \forall y.\neg Pny$ of EM_1 (with P atomic), we may formulate an hypothesis about which side of the disjunction is true. If we know that Pnm is true for

some m, we know that $\exists y.Pny$ is true. Otherwise we may assume $\forall y.\neg Pny$ as hypothesis, because it is a falsifiable hypothesis.

We formalize the process of making hypotheses about EM_1 by a finite state of knowledge, called S, collecting the instances Pnm which we know to hold, e.g. by direct calculation. If we have evidence that Pnm holds for some m (that is, $Pnm \in S$) we know that $\exists y Pny$ is true; in the other case, we assume that $\forall y \neg Pny$ is true. So S defines a set of hypotheses on EM_1, of the form $\forall y \neg Pny$: universal falsifiable statements. Using S a realizer r may effectively decide which side of a given instance of EM_1 is true, at the price of making mistakes: to decide if $\forall y \neg Pny$ is true, r looks for any Pnm in the finite state S and outputs "false" if the research is successful, "true" otherwise. If and when from an hypothesis $\forall y \neg Pny$ we obtain some false conclusion $\neg Pnm$, the realizer r returns the additional knowledge: "Pnm is true", to be added to S.

Extending Berardi-de' Liguoro interpretation to $\mathsf{HA} + \mathsf{EM}_1$. In our chapter, we interpret each classical proof p of A in $\mathsf{HA} + \mathsf{EM}_1$ by a "learning realizer" r. r returns a "prediction" of the truth of this formula, based on the information in S, and some additional knowledge in the case the prediction is effectively falsified. For example, in front of a formula $\exists x.A \wedge B$, a realizer r predicts that $A(n) \wedge B(n)$ is true for some numeral n (and since n depends on S, in our model we change the notion of individual, interpreting "numbers" as computable maps from the set of bases of knowledge to $\mathbb{N}$). Then r predicts, say, that $B(n)$ is true, and so on, until r arrives at some atomic formula, say $\neg Pnm$. Either Pnm is actually true, or r is able to effectively find one or more flawed hypothesis $\forall x.\neg Q_1 n_1 x, \ldots, \forall x.\neg Q_k n_k x$ among the hypotheses used to predict that Pnm is true, and for each flawed hypothesis one counterexample $Q_1 n_1 m_1, \ldots, Q_k n_k m_k$. In this case, r requires to enlarge our state of knowledge S by including the information "$Q_1 n_1 m_1$ is true", ..., "$Q_k n_k m_k$ is true".

Our Interactive Realizability differs from Intuitionistic Realizability in the notion of individual (the value of an individual may depend on our knowledge state), and in the realizability relation for the atomic case. In our interpretation, to realize an atomic formula does not mean that the formula is true, but that the realizer requires to extend our state of knowledge S if the formula is not true. The realizer is thought as a learning device. Each extension of S may change the value of the individuals which are parameters of the atomic formula, and therefore may make the atomic formula false again. Then the realizer requires to extend S again, and so forth. The convergence of this "interaction" between a realizer and a group of individuals follows by Avigad's fixed point thm. [5] (a constructive proof may be found in [7]), and it is the analogue of the termination of Hilbert's ϵ-substitution method.

Why the Arithmetic $\mathsf{HA} + \mathsf{EM}_1$ instead of considering the full Peano Arithmetic? We have two main reasons. First, we observe that EM_1 enjoys a very good property: the information about its truth can be computed in the limit, in the sense of Gold [22], as we saw en passant when discussing Hayashi's realizability. This implies that witnesses for existential and disjunctive statements too can be learned in the limit, as shown in Hayashi [28]. In chapter 4 we show that realizers which we will be able to extract from proofs have a straightforward interpretation as winning strategies in 1-Backtracking games [9], which are the most natural and simple instances of Coquand's style games. Secondly, a great deal of mathematical theorems are proved by using EM_1 alone ([1], [8]). Third, as shown in chapter 5, already $\mathsf{HA} + \mathsf{EM}_1$ - plus Gödel's double negation translation - suffices to interpret all

provably total recursive functions of PA, with the advantage of eliminating the extra step of Friedman or Dialectica translation (see Kohlenbach for [32] these latter)

Plan of the Chapter. The chapter is organized as follows. In §3.2 we define the term calculus in which our realizers will be written: a version of Gödel's system T, extended with some syntactic sugar, in order to represent bases of knowledge (which we shall call states) and to manipulate them. Then we prove a convergence property for this calculus (as in Avigad [5] or in [7]). In §3.3, we introduce the notion of realizability and prove our Main Theorem, the Adequacy Theorem: "if a closed arithmetical formula is provable in $\mathsf{HA}+\mathsf{EM}_1$, then it is realizable". In §3.4 we conclude the discussion about our notion of realizability by comparing it with other notions of realizability for classical logic, then we consider some possible future work.

3.2. The Term Calculus

In this section we formalize the intuition of "learning realizer" we discussed in the introduction.

We associate to any instance $\exists yPxy \vee \forall y\neg Pxy$ of EM_1 (Excluded Middle restricted to Σ^0_1-formulas) two functions χ_P and φ_P. The function χ_P takes a knowledge state S, a numeral n, and returns a guess for the truth value of $\exists y.Pny$. When this guess is "true" the function φ_P returns a witness m of $\exists y.Pny$. The guess for the truth value of $\exists y.Pny$ is computed w.r.t. the knowledge state S, and it may be wrong. For each constant s denoting some knowledge state S, the function $\lambda x^{\mathtt{N}}\ \chi_P(s,x)$ is some "approximation" of an ideal map $\lambda x^{\mathtt{N}}\ \mathsf{X}_P(x)$, the *oracle* returning the truth value of $\exists y.Pxy$. In the same way, the function $\lambda x^{\mathtt{N}}\ \phi_P(s,x)$ is some "approximation" of an ideal map $\lambda x^{\mathtt{N}}\ \Phi_P(x)$, the *Skolem map* for $\exists y.Pxy$, returning some y such that Pxy if any, and 0 otherwise. The Skolem axioms effectively used by a given proof take the place of a set of experiments testing the correctness of the predictions made by $\varphi_P(s,x), \chi_P(s,x)$ about $\mathsf{X}_P(x), \Phi_P(x)$ (we do not check the correctness of φ_P, χ_P in an exhaustive way, but only on the values required by the Skolem axioms used by a proof).

Our Term Calculus is an extension of Gödel's system T (see chapter 2 or [21]). From now on, if t, u are terms of T with $t = u$ we denote provable equality in T. If $k \in \mathbb{N}$, the numeral denoting k is the closed normal term $\overline{k} = \mathsf{S}^k(0)$ of type $\mathtt{N}$. All closed normal terms of type $\mathtt{N}$ are numerals (see chapter 2). We recall that any closed normal term of type $\mathtt{Bool}$ in T is $\mathtt{True}$ or $\mathtt{False}$.

We introduce a notation for ternary projections: if $T = A \times (B \times C)$, with p_0, p_1, p_2 we respectively denote the terms π_0, $\lambda x : T.\pi_0(\pi_1(x))$, $\lambda x : T.\pi_1(\pi_1(x))$. If $u = \langle u_0, \langle u_1, u_2\rangle\rangle : T$, then $p_i u = u_i$ in T for $i = 0, 1, 2$. We abbreviate $\langle u_0, \langle u_1, u_2\rangle\rangle : T$ with $\langle u_0, u_1, u_2\rangle : T$. We formalize the idea of "finite information about EM_1" by the notion of *state of knowledge*.

Definition 3.2.1 (States of Knowledge and Consistent Union). We define:

(1) A k-ary *predicate* of T is any closed normal term $P : \mathtt{N}^k \to \mathtt{Bool}$ of T.

(2) An atom is any triple $\langle P, \vec{n}, m\rangle$, where P is a $(k+1)$-ary predicate, $\vec{n}, m$ are $k+1$ numerals, and $P\vec{n}m = \mathtt{True}$ in T.

(3) Two atoms $\langle P, \vec{n}, m\rangle$, $\langle P', \vec{n}', m'\rangle$ are *consistent* if $P = P'$ and $\vec{n} = \vec{n}'$ imply $m = m'$.

(4) A state of knowledge, shortly a *state*, is any finite set S of pairwise consistent atoms.

(5) Two states S_1, S_2 are consistent if $S_1 \cup S_2$ is a state.

(6) $\mathbb{S}$ is the set of all states of knowledge.

(7) The *consistent union* $S_1 \mathcal{U} S_2$ of $S_1, S_2 \in \mathbb{S}$ is $S_1 \cup S_2 \in \mathbb{S}$ minus all atoms of S_2 which are inconsistent with some atom of S_1.

We think of an atom $\langle P, \vec{n}, m\rangle$ as the code of a witness for $\exists y.P(\vec{n}, y)$. Consistency condition allows at most one witness for each $\exists y.P(\vec{n}, y)$ in each knowledge state S. Two states S_1, S_2 are consistent if and only if each atom of S_1 is consistent with each atom of S_2.

$S_1 \mathcal{U} S_2$ is an non-commutative operation: whenever an atom of S_1 and an atom of S_2 are inconsistent, we arbitrarily keep the atom of S_1 and we reject the atom of S_2, therefore for some S_1, S_2 we have $S_1 \mathcal{U} S_2 \neq S_2 \mathcal{U} S_1$. $\mathcal{U}$ is a "learning strategy", a way of selecting a consistent subset of $S_1 \cup S_2$. It is immediate to show that $\mathcal{U}$ is an associative operation on the set of consistent states, with neutral element $\emptyset$, with upper bound $S_1 \cup S_2$, and returning a non-empty state whenever $S_1 \cup S_2$ is non-empty.

LEMMA 3.2.2 . *Assume $i \in \mathbb{N}$ and $S_1, \ldots, S_i \in \mathbb{S}$.*

(1) $S_1 \mathcal{U} \ldots \mathcal{U} S_i \subseteq S_1 \cup \ldots \cup S_i$
(2) $S_1 \mathcal{U} \ldots \mathcal{U} S_i = \emptyset$ *implies* $S_1 = \ldots = S_i = \emptyset$.

In fact, the whole realizability Semantics is a Monad [12]. In [12], it is proved that our realizability Semantics is parametric with respect to the definition we choose for $\mathcal{U}$. Any associative operation $\mathcal{U}$, with neutral element $\emptyset$ and satisfying the two properties of Lemma 3.2.2, defines a different but sound realizability Semantics, corresponding to a different "learning strategy". An immediate consequence of Lemma 3.2.2 is:

LEMMA 3.2.3 . *Assume $S, S_1, S_2 \in \mathbb{S}$.*

(1) *If S is consistent with S_1, S_2, then S is consistent with $S_1 \mathcal{U} S_2$.*
(2) *If S is disjoint with S_1, S_2, then S is disjoint with $S_1 \mathcal{U} S_2$.*

For each state of knowledge S we assume having a unique constant s denoting it. We denote the state denoted by a constant s with $|s|$ and as usual with $|\_|^{-1}$ the inverse of $|\_|$; that is, $||s||^{-1} = s$. We assume $\varnothing$ is the state constant denoting the empty state $\emptyset$; that is, $|\varnothing| = \emptyset$. We define with

$$\mathcal{T}_{\mathtt{S}} = \mathsf{T} + \mathtt{S} + \{s \mid |s| \in \mathbb{S}\}$$

the extension of T with one atomic type $\mathtt{S}$ denoting $\mathbb{S}$, and a constant s for each state $S \in \mathbb{S}$, and *no* new reduction rule. We denote states by $S, S', \ldots$ and state constants by $s, s', \ldots$. Any closed normal form of type $\mathtt{N}, \mathtt{Bool}, \mathtt{S}$ in $\mathcal{T}_{\mathtt{S}}$ is, respectively, some numeral n, some boolean $\mathtt{True}, \mathtt{False}$, some state constant s. Computation on states will be defined by some suitable set of algebraic reduction rules we call "functional".

DEFINITION 3.2.4 (FUNCTIONAL SET OF RULES). Let C be any set of constants, each one of some type $A_1 \to \ldots \to A_n \to A$, for some $A_1, \ldots, A_n, A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{S}\}$. We say that $\mathcal{R}$ is a *functional set of reduction rules* for C if $\mathcal{R}$ consists, for all $c \in C$ and all $a_1 : A_1, \ldots, a_n : A_n$ closed normal terms of $\mathcal{T}_\mathtt{S}$, of exactly one rule $ca_1 \ldots a_n \mapsto a$, for some closed normal term $a : A$ of $\mathcal{T}_\mathtt{S}$.

THEOREM 3.2.5 . *Assume that $\mathcal{R}$ is a functional set of reduction rules for C (def. 3.2.4). Then $\mathcal{T}_\mathtt{S} + C + \mathcal{R}$ enjoys: i) strong normalization; ii) weak-Church-Rosser (uniqueness of normal forms) for all closed terms of atomic types.*

PROOF. *(Sketch)* For strong normalization, see Berger [13] (the constants $s : \mathtt{S}$ and $c \in C$ are trivially strongly computable). For weak Church-Rosser property, we start from the fact that there is the canonical set-theoretical model $\mathcal{M}$ of $\mathcal{T}_\mathtt{S}+C+\mathcal{R}$. The interpretation of $\mathtt{Bool}, \mathtt{N}, \mathtt{S}$ in $\mathcal{M}$ consists of all closed normal form of these types. Arrows and pairs are interpreted set-theoretically. Each constant $c \in C$ is interpreted by some map f_c, defined by $f_c(a_1, \ldots, a_n) = a$ for all reduction rules $(ca_1 \ldots a_n \mapsto a) \in \mathcal{R}$. Assume $u, v : A$ are closed normal term, $A = \mathtt{Bool}, \mathtt{N}$, or $\mathtt{S}$ is an atomic type, and u, v are equal in $\mathcal{T}_\mathtt{S} + C + \mathcal{R}$, in order to prove that u, v are the same term. u, v are equal in $\mathcal{M}$ because $\mathcal{M}$ is a model of $\mathcal{T}_\mathtt{S} + C + \mathcal{R}$. By induction on w we prove that if w is a closed normal form of atomic type $\mathcal{T} + C + \mathcal{R}$, then w is a numeral, or $\mathtt{True}, \mathtt{False}$, or a state constant, and therefore w is interpreted by itself in $\mathcal{M}$. From u, v equal in $\mathcal{M}$ we conclude that u, v are the same term of $\mathcal{T}_\mathtt{S} + C + \mathcal{R}$.

□

We define two extensions of $\mathcal{T}_\mathtt{S}$: an extension $\mathcal{T}_{\text{Class}}$ with symbols denoting the non-computable maps $\mathsf{X}_P, \mathsf{\Phi}_P$ and no computable reduction rules, another extension $\mathcal{T}_{\text{Learn}}$, with the computable approximations χ_P, ϕ_P of $\mathsf{X}_P, \mathsf{\Phi}_P$, and a computable set of reduction rules. We use the elements of $\mathcal{T}_{\text{Class}}$ to represent non-computable realizers, and the elements of $\mathcal{T}_{\text{Learn}}$ to represent a computable "approximation" of a realizer. In the next definition, we denote terms of type $\mathtt{S}$ by $\rho, \rho', \ldots$.

DEFINITION 3.2.6 . Assume $P : \mathtt{N}^{k+1} \to \mathtt{Bool}$ is a $k+1$-ary predicate of T. We introduce the following constants:

(1) $\mathsf{X}_P : \mathtt{N}^k \to \mathtt{Bool}$ and $\mathsf{\Phi}_P : \mathtt{N}^k \to \mathtt{N}$.

(2) $\chi_P : \mathtt{S} \to \mathtt{N}^k \to \mathtt{Bool}$ and $\varphi_P : \mathtt{S} \to \mathtt{N}^k \to \mathtt{N}$.

(3) $\Cup : \mathtt{S} \to \mathtt{S} \to \mathtt{S}$.

(4) $\mathsf{Add}_P : \mathtt{N}^{k+1} \to \mathtt{S}$ and $\mathsf{add}_P : \mathtt{S} \to \mathtt{N}^{k+1} \to \mathtt{S}$.

We denote $\Cup\rho_1\rho_2$ with $\rho_1 \Cup \rho_2$.

(1) $\Xi_\mathtt{S}$ is the set of all constants $\chi_P, \varphi_P, \Cup, \mathsf{add}_P$.

(2) Ξ is the set of all constants $\mathsf{X}_P, \mathsf{\Phi}_P, \Cup, \mathsf{Add}_P$.

(3) $\mathcal{T}_{\text{Class}} = \mathcal{T}_{\mathsf{S}} + \Xi$.

(4) A term $t \in \mathcal{T}_{\text{Class}}$ has state $\varnothing$ if it has no state constant different from $\varnothing$.

Let $\vec{t} = t_1 \dots t_k$. We interpret $\chi_P s\vec{t}$ and $\varphi_P s\vec{t}$ respectively as a "guess" for the values of the oracle and the Skolem map X_P and Φ_P for $\exists y.P\vec{t}y$, guess computed w.r.t. the knowledge state denoted by the constant s. There is no set of computable reduction rules for the constants $\Phi_P, \mathsf{X}_P \in \Xi$, and therefore no set of computable reduction rules for $\mathcal{T}_{\text{Class}}$.

If s_1, s_2 are state constants, we interpret $s_1 \Cup s_2$ as denoting the consistent union $|s_1|\mathcal{U}|s_2|$. Add_P denotes the map constantly equal to the empty state $\emptyset$. $\mathsf{add}_P s\vec{n}m$ denotes the empty state $\emptyset$ if we cannot add the atom $\langle P, \vec{n}, m\rangle$ to $|s|$, either because $\langle P, \vec{n}, l\rangle \in |s|$ for some numeral l, or because $P\vec{n}m = \mathtt{False}$; $\mathsf{add}_P s\vec{n}m$ denotes the state $\{\langle P, \vec{n}, m\rangle\}$ otherwise. We define a system $\mathcal{T}_{\text{Learn}}$ with reduction rules over Ξ_{S} by a functional reduction set $\mathcal{R}_{\mathsf{S}}$.

DEFINITION 3.2.7 (THE SYSTEM $\mathcal{T}_{\text{LEARN}}$). Let s, s_1, s_2 be state constants. Let $\langle P, \vec{n}, m\rangle$ be an atom. $\mathcal{R}_{\mathsf{S}}$ is the following functional set of reduction rules for Ξ_{S}:

$$\chi_P s\vec{n} \mapsto \begin{cases} \mathtt{True} & \text{if } \exists m.\ \langle P, \vec{n}, m\rangle \in |s| \\ \mathtt{False} & \text{otherwise} \end{cases}$$

$$\varphi_P s\vec{n} \mapsto \begin{cases} m & \text{if } \exists m.\ \langle P, \vec{n}, m\rangle \in |s| \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{add}_P s\vec{n}m \mapsto \begin{cases} \varnothing & \text{if } \exists l.\ \langle P, \vec{n}, l\rangle \in |s| \vee P\vec{n}m = \mathtt{False} \\ |\{\langle P, \vec{n}, m\rangle\}|^{-1} & \text{otherwise} \end{cases}$$

$$s_1 \Cup s_2 \mapsto s_3, \text{ where } s_3 \text{ is the state constant such that } |s_3| = |s_1|\mathcal{U}|s_2|$$

We define $\mathcal{T}_{\text{Learn}} = \mathcal{T}_{\mathsf{S}} + \Xi_{\mathsf{S}} + \mathcal{R}_{\mathsf{S}}$.

Remark. $\mathcal{T}_{\text{Learn}}$ is nothing but $\mathcal{T}_{\mathsf{S}}$ with some "syntactic sugar". By Theorem 3.2.5, $\mathcal{T}_{\text{Learn}}$ is strongly normalizing and has the weak Church-Rosser property for closed term of atomic types. $\mathcal{T}_{\text{Learn}}$ satisfies a Normal Form Property.

LEMMA 3.2.8 (NORMAL FORM PROPERTY FOR $\mathcal{T}_{\text{LEARN}}$). *Assume A is either an atomic type or a product type. Then any closed normal term $t \in \mathcal{T}_{Learn}$ of type A is: a numeral $n : \mathtt{N}$, or a boolean $\mathtt{True}, \mathtt{False} : \mathtt{Bool}$, or a state constant $s : \mathtt{S}$, or a pair $\langle u, v\rangle : B \times C$.*

PROOF. *(Sketch)* By induction over t. For some $\vec{v}$, either t is $(\lambda\vec{x}.u)(\vec{v})$, or t is $\langle u, w\rangle(\vec{v})$, or t is $x(\vec{v})$ for some variable x, or t is $c(\vec{v})$ for some constant c, and either $c = 0, \mathrm{S}, \mathtt{True}, \mathtt{False}, s$, $R_T, \mathtt{if}_T, \pi_i$ is some constant of $\mathcal{T}_{\mathsf{S}}$, or $c \in \Xi_{\mathsf{S}}$. If $t = (\lambda\vec{x}.u)(\vec{v})$, then t has an arrow type if $\vec{v} = \emptyset$, while t is not normal if $\vec{v} \neq \emptyset$. If $t = \langle u, w\rangle(\vec{v})$, then $\vec{v} = \emptyset$ and we are done. If

$t = x(\vec{v})$ then t is not closed. The only case left is $t = c(\vec{u}) : A$. A is not an arrow type, therefore all arguments of c are in $\vec{u}$. If $t = 0$ we are done, if $t = \mathrm{S}(u)$ we apply the induction hypothesis, if $t = \mathtt{True}, \mathtt{False} : \mathtt{Bool}$ or $t = s : \mathtt{S}$ or $t = \langle u, v\rangle$ we are done. Otherwise either $t = R_T(n, f, a)\vec{t}, \mathtt{if}_T(b, a_1, a_2)\vec{t}, \pi_i(v)\vec{t}$, or $t = \chi_P(u, \vec{w}) : \mathtt{N}$, or $t = \varphi_P(u, \vec{w}) : \mathtt{N}$, or $t = \Cup(u_1, u_2) : \mathtt{S}$, or $t = \mathsf{add}_P(u, \vec{w}) : \mathtt{S}$. The proper subterms $n, w_1, \ldots, w_k : \mathtt{N}$, $b : \mathtt{Bool}$, $v : A \times B$, $u, u_1, u_2 : \mathtt{S}$ of t have atomic or product type and are closed normal. By induction hypothesis they are, respectively, a numeral, a boolean, a pair, a state constant. In all cases, t is not normal.

□

Let $t_1 t_2 \in \mathcal{T}_{\mathrm{Learn}}$ be two closed terms of type $\mathtt{S}$. We abbreviate "the normal forms of t_1, t_2 denote two states which are consistent and disjoint" by: t_1, t_2 are consistent and disjoint. $\varnothing, s$ are consistent and disjoint for every state constant s. The maps denoted by $\Cup, \mathsf{add}_P$ preserve the relation "to be consistent and disjoint".

LEMMA 3.2.9 . *Assume s, s_1, s_2 are state constants and $\langle P, \vec{n}, m\rangle$ is an atom.*

(1) *$s, (\mathsf{add}_P s\vec{n}m)$ are consistent and disjoint.*

(2) *Assume s, s_1 are consistent and disjoint, and s, s_2 are consistent and disjoint. Then $s, s_1 \Cup s_2$ are consistent and disjoint.*

PROOF. (1) If $\mathsf{add}_P s\vec{n}m$ denotes the empty state the thesis is immediate. Otherwise $\mathsf{add}_P s\vec{n}m$ denotes $\{\langle P, \vec{n}, m\rangle\}$ and $\langle P, \vec{n}, l\rangle \notin |s|$ for all numerals l. Then $\{\langle P, \vec{n}, m\rangle\}$ is consistent and disjoint with $|s|$.
(2) By Lemma 3.2.3.

□

Each (in general, non-computable) term $t \in \mathcal{T}_{\mathrm{Class}}$ is associated to a set $\{t[s] \mid s \text{ is a state constant}\} \subseteq \mathcal{T}_{\mathrm{Learn}}$ of computable terms we call its "approximations", one for each state constant s.

DEFINITION 3.2.10 (APPROXIMATION AT STATE s). Assume $t \in \mathcal{T}_{\mathrm{Class}}$ and s is a state constant. We call "approximation of t at state s" the term $t[s]$ of $\mathcal{T}_{\mathrm{Learn}}$ obtained from t by replacing each constant X_P with $\chi_P s$, each constant Φ_P with $\varphi_P s$, each constant Add_P with $\mathsf{add}_P s$.

We interpret any $t[s] \in \mathcal{T}_{\mathrm{Learn}}$ as a learning process evaluated w.r.t. the information taken from a state constant s (the same s for the whole term).

Assume $t \in \mathcal{T}_{\mathrm{Class}}$ is closed, $t : \mathtt{S}$ and s is a state constant. Then $t[s]$ is a closed term of $\mathcal{T}_{\mathrm{Learn}}$, and its normal form, by the Normal Form Property 3.2.8, is some state constant s'. We conclude $t[s] = s'$ in $\mathcal{T}_{\mathrm{Learn}}$. We prove that s, s' are consistent and disjoint.

LEMMA 3.2.11 . *Assume s is a state constant, $t \in \mathcal{T}_{Class}$, $t : \mathtt{S}$ is closed, and all state constants in t are consistent and disjoint with s.*

(1) *If $t[s]$ reduces to $t'[s]$, then all state constants in t' are consistent and disjoint with s.*

(2) $s, t[s]$ *are consistent and disjoint.*

(3) *If* $u \in \mathcal{T}_{Class}$, $u : \mathtt{S}$ *and all state constants in* u *are* $\varnothing$, *then* $s, u[s]$ *are consistent and disjoint.*

PROOF. (1) It is enough to consider a one-step reduction. Suppose that $t[s]$ reduces to $t'[s]$ by contraction of a redex r of $t[s]$. If r is $(\lambda x u)t$ or $\mathsf{R}_T uv\mathsf{S}(w)$ or $\mathtt{if}_T(b, a_1, a_2)$ or $\pi_i\langle v_1, v_2\rangle$ or $\chi_P s\vec{n}$, or $\varphi_P s\vec{n}$, then its contractum r' does not contain any new state constant; hence, all state constants in t' are consistent and disjoint with s. If r is $s_1 \Cup s_2$ or $\mathtt{add}_P s\vec{n}m$, then both s, s_1 and s, s_2 are consistent and disjoint state constants by hypothesis on t; therefore, by Lemma 3.2.9, in both cases s and the contraction of r are consistent and disjoint; so all state constants in t' are consistent and disjoint with s.

(2) Every reduct of $t[s]$ is $t'[s]$ for some $t' \in \mathcal{T}_{\text{Class}}$. If $t[s]$ reduces to a normal form $t'[s] \equiv s'$, then the only possibility is $t' \equiv s'$. By the previous point 1, we conclude that s' is consistent and disjoint with s.

(3) By the previous point 2, and the fact that the only state constant $\varnothing$ in u is consistent and disjoint with any s.

□

We introduce now a notion of convergence for families of terms $\{t[s_i]\}_{i\in\mathbb{N}} \subseteq \mathcal{T}_{\text{Learn}}$, defined by some $t \in \mathcal{T}_{\text{Class}}$ and indexed over a set of state constants $\{s_i\}_{i\in\mathbb{N}}$. Informally, "t convergent" means that the normal form of $t[s]$ eventually stops changing when the knowledge state s increases. If s_1, s_2 are state constants, we write $s_1 \leq s_2$ for $|s_1| \subseteq |s_2|$. We say that a sequence $\{s_i\}_{i\in\mathbb{N}}$ of state constants is a weakly increasing chain of states (is w.i. for short), if $s_i \leq s_{i+1}$ for all $i \in \mathbb{N}$.

DEFINITION 3.2.12 (CONVERGENCE). Assume that $\{s_i\}_{i\in\mathbb{N}}$ is a w.i. sequence of state constants, and $u \in \mathcal{T}_{\text{Class}}$.

(1) u converges in $\{s_i\}_{i\in\mathbb{N}}$ if $\exists i \in \mathbb{N}.\forall j \geq i.u[s_j] = u[s_i]$ in $\mathcal{T}_{\text{Learn}}$.

(2) u converges if u converges in every w.i. sequence of state constants.

We remark that if u is convergent, we do not ask that u is convergent to the *same* value on *all* w.i. chain of states. The value learned by u may depend on the information contained in the particular chain of state constants by which u gets the knowledge. The chain of states, in turn, is selected by the particular definition we use for the "learning strategy" $\mathcal{U}$. Different "learning strategies" may learn different values.

THEOREM 3.2.13 (STABILITY THEOREM). *Assume* $t \in \mathcal{T}_{Class}$ *is a closed term of atomic type* A ($A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{S}\}$). *Then* t *is convergent.*

PROOF. *(Classical).* Assume S is any consistent and possibly infinite set of atoms. We define some (in general, *not* computable) functional reduction set $\mathcal{R}(S)$ for the set Ξ of

constants and for $\mathcal{T}_{\text{Class}}$. The reductions for $\mathsf{X}_P, \Phi_P, \mathsf{Add}_P$ are those for $\chi_P, \phi_P, \mathsf{add}_P$ in $\mathcal{T}_{\text{Learn}}$:

(1) If $\langle P, \vec{n}, m\rangle \in S$, then $(\mathsf{X}_P\vec{n} \mapsto \mathtt{True}), (\Phi_P\vec{n} \mapsto m) \in \mathcal{R}(S)$, else $(\mathsf{X}_P\vec{n} \mapsto \mathtt{False}), (\Phi_P\vec{n} \mapsto 0) \in \mathcal{R}(S)$.

(2) $\mathsf{Add}_P\vec{n}m \mapsto \varnothing \in \mathcal{R}(S)$ if either $\langle P, \vec{n}, l\rangle \in S$ for some numeral l or $P\vec{n}m = \mathtt{False}$; otherwise, $\mathsf{Add}_P\vec{n}m \mapsto |\{\langle P, \vec{n}, m\rangle\}|^{-1} \in \mathcal{R}(S)$.

and the reduction for $\Cup$ in $\mathcal{R}(S)$ is the reduction for $\Cup$ in $\mathcal{T}_{\text{Learn}}$. By theorem 3.2.5, $\mathcal{T}_{\text{Class}} + \mathcal{R}(S)$ is strongly normalizing and weak-CR for all closed terms of atomic type, for any consistent set of atoms S. For the rest of the proof, let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of state constants. Assume $t \in \mathcal{T}_{\text{Class}}$ is a closed term of atomic type A.

Claim. For any state constant s, the map $u \mapsto u[s]$ is a bijection from the reduction tree of t in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s|)$ to the reduction tree of $t[s]$ in $\mathcal{T}_{\text{Learn}}$.

Proof of the Claim. By induction over the reduction tree of $t[s]$. Every reduction $\beta, \pi, \mathtt{if}_T, R_T, \Cup$ over $t[s]$ may be obtained from the same reduction over t. All occurrences of $\chi_P, \varphi_P, \mathsf{add}_P$ in the reduction tree of $t[s]$ are of the form $\chi_P s, \varphi_P s, \mathsf{add}_P s$, therefore every reduction over $\chi_P, \varphi_P, \mathsf{add}_P$ may be obtained from the corresponding reduction over $\mathsf{X}_P, \Phi_P, \mathsf{Add}_P$.

Assume now a is the (unique, by weak-CR) normal form of t in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s|)$. By the *Claim*, $a[s]$ is the normal form of $t[s]$ in $\mathcal{T}_{\text{Learn}}$. Since a is normal in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s|)$, there is no $\mathsf{X}_P, \Phi_P, \mathsf{Add}_P$ in a. Thus a and $a[s]$ are the same term: t and $t[s]$ have the same normal form respectively in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s|)$ and in $\mathcal{T}_{\text{Learn}}$. Let $\{s_i\}_{i\in\mathbb{N}}$ be a given sequence of state constants. Define $S_\omega = \cup_{i\in\mathbb{N}}|s_i|$. By strong normalization, the reduction tree of t in $\mathcal{T}_{\text{Class}} + \mathcal{R}(S_\omega)$ is finite. Therefore in this reduction tree are used only *finitely many* reduction rules from $\mathcal{R}(S_\omega)$, and for some numeral n it is equal to the reduction tree of t in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s_n|)$, and in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s_m|)$ for all $m \geq n$. We deduce that for all $m \geq n$ the normal forms of t in $\mathcal{T}_{\text{Class}} + \mathcal{R}(|s_m|)$ are the same. Thus, the normal form in $\mathcal{T}_{\text{Learn}}$ of all $t[s_m]$ with $m \geq n$ are the same, as we wished to show. □

Remark 3.2.14 . The idea of the proof of theorem 3.2.13 corresponds exactly to the intuition of the introduction. During any computation, the oracles X_P and Φ_P are consulted a finite number of times and hence asked for a finite number of values. When our state of knowledge is great enough, we can substitute the oracles with their approximation $\chi_P s$ and $\varphi_P s$ for some state constant s, and we will obtain the same oracle values and hence the same results.

The proof, though non constructive, is short and explains well why the result is true. However, provided we replace the notion of convergence used in this chapter with the intuitionistic notion introduced in [7], we are able to reformulate and prove theorem 3.2.13 in a purely intuitionistic way, achieving thus a constructive description of learning in $\mathsf{HA}{+}\mathsf{EM}_1$. Being the intuitionistic proof much more elaborated and less intuitive than the present one and connected with other foundationally interesting results, it will be the subject of chapter 5.

Our proof of convergence follows the pattern of Avigad's one in [5]. A closed term $t \in \mathcal{T}_{\text{Class}}$ of atomic type and in the constants $c_1, \ldots, c_n \in \Xi$, may be seen as a functional F_t which maps functions $f_1, \ldots, f_n$ of the same type of $c_1, \ldots c_n$ into an object of atomic type: $F_t(f_1, \ldots, f_n)$ is defined as the normal form of t in $\mathcal{T}_{\text{Class}} + \mathcal{R}$, where $\mathcal{R} = \{c_i a_1 \ldots a_n \mapsto a \mid f_i(a_1, \ldots, a_n) = a \text{ and } i \in \{1, \ldots, n\}\}$. F_t is continuous in the sense of Avigad. Moreover,

since X_P and Add_P have a set-theoretical definition in terms of Φ_P, we may assume F_t depends only on the functions which define in $\mathcal{R}$ the reduction rules for $\Phi_{P_1}, \dots \Phi_{P_n}$. Then, if t is of type $\mathtt{S}$, it is not difficult to see that F_t represents an update procedure with respect to any of its argument. The fact that F_t is an update procedure implies convergence for t and the zero theorem 3.2.15.

As last result of this section, we prove that if we start from any state constant s and we repeatedly apply any closed term $t : \mathtt{S}$ of $\mathcal{T}_{\text{Class}}$ of state $\varnothing$ (see definition 3.2.6), we obtain a "zero" of t, that is a state constant s_n such that $t[s_n] = \varnothing$. We interpret this by saying that any term t represents a terminating learning process.

THEOREM 3.2.15 (ZERO THEOREM). *Let $t : \mathtt{S}$ be a closed term of $\mathcal{T}_{Class}$ of state $\varnothing$ and s any state constant. Define, by induction on n, a sequence $\{s_n\}_{n\in\mathbb{N}}$ of state constants such that: $s_0 = s$ and $s_{n+1} = s_n \Cup t[s_n]$. Then, there exists an n such that $t[s_n] = \varnothing$.*

PROOF. $s_0, s_1, s_2, \dots$ is a weakly increasing chain of state constants by construction. By theorem 3.2.13, t converges over this chain: there exists $k \in \mathbb{N}$ such that for every $j \geq k$, $t[s_j] = t[s_k]$. By choice of k

$$\begin{aligned} s_{k+2} &= s_{k+1} \Cup t[s_{k+1}] \\ &= (s_k \Cup t[s_k]) \Cup t[s_{k+1}] \\ &= (s_k \Cup t[s_k]) \Cup t[s_k] \\ &= s_k \Cup t[s_k] \\ &= s_{k+1} \end{aligned}$$

Since $s_{k+2} = s_{k+1}$ and $s_{k+1}, t[s_{k+1}]$ are consistent and disjoint by lemma 3.2.11, we conclude $t[s_{k+1}] = \varnothing$.

□

3.3. An Interactive Learning-Based Notion of Realizability

In this section we introduce the notion of realizability for $\mathsf{HA}+\mathsf{EM}_1$, Heyting Arithmetic plus Excluded Middle on Σ^0_1-formulas, then we prove our main Theorem, the Adequacy Theorem: *"if a closed arithmetical formula is provable in $\mathsf{HA}+\mathsf{EM}_1$, then it is realizable"*.

We first define the formal system $\mathsf{HA}+\mathsf{EM}_1$, from now on "Extended EM_1 Arithmetic". We represent atomic predicates of $\mathsf{HA}+\mathsf{EM}_1$ with (in general, non-computable) closed terms of $\mathcal{T}_{\text{Class}}$ of type $\mathtt{Bool}$. Terms of $\mathsf{HA}+\mathsf{EM}_1$ may include function symbols X_P, Φ_P denoting non-computable functions: oracles and Skolem maps for Σ^0_1-formulas $\exists x.P x\vec{n}$, with P predicate of T. We remark that our realizability can be formulated already for the standard language of Arithmetic: we add non computable functions to the language for greater generality. We assume having in T some terms $\Rightarrow_{\mathtt{Bool}} : \mathtt{Bool}, \mathtt{Bool} \to \mathtt{Bool}, \neg_{\mathtt{Bool}} : \mathtt{Bool} \to \mathtt{Bool}, \dots$, implementing boolean connectives. If $t_1, \dots, t_n, t \in \mathsf{T}$ have type $\mathtt{Bool}$ and are made from free variables all of type $\mathtt{Bool}$, using boolean connectives, we say that t is a tautological consequence of $t_1, \dots, t_n$ in T (a tautology if $n = 0$) if all boolean assignments making $t_1, \dots, t_n$ equal to $\mathtt{True}$ in T also make t equal to $\mathtt{True}$ in T.

DEFINITION 3.3.1 (EXTENDED EM_1 INTUITIONISTIC ARITHMETIC: $\mathsf{HA}+\mathsf{EM}_1$). The language $\mathcal{L}_{\text{Class}}$ of $\mathsf{HA}+\mathsf{EM}_1$ is defined as follows.

(1) The terms of $\mathcal{L}_{\text{Class}}$ are all $t \in \mathcal{T}_{\text{Class}}$ with state $\varnothing$, such that $t : \mathtt{N}$ and $FV(t) \subseteq \{x_1^{\mathtt{N}}, \dots, x_n^{\mathtt{N}}\}$ for some $x_1, \dots, x_n$.

(2) The atomic formulas of $\mathcal{L}_{\text{Class}}$ are all $Qt_1 \dots t_n \in \mathcal{T}_{\text{Class}}$, for some $Q : \mathtt{N}^n \to \mathtt{Bool}$ *closed term of* $\mathcal{T}_{Class}$ of state $\varnothing$, and some terms $t_1, \dots, t_n$ of $\mathcal{L}_{\text{Class}}$.

(3) The formulas of $\mathcal{L}_{\text{Class}}$ are built from atomic formulas of $\mathcal{L}_{\text{Class}}$ by the connectives $\vee, \wedge, \to \forall, \exists$ as usual.

A formula of HA is a formula of $\mathsf{HA}+\mathsf{EM}_1$ in which all predicates and terms are terms of T.

Deduction rules for $\mathsf{HA}+\mathsf{EM}_1$ are as in van Dalen [15], with: *(i)* an axiom schema for EM_1; *(ii)* the induction rule; *(iii)* as Post rules: all axioms of equality and ordering on $\mathtt{N}$, all equational axioms of T, and one schema for each tautological consequences of T. *(iv)* the axiom schemas for oracles: $P(\vec{t}, t) \Rightarrow_{\mathtt{Bool}} \mathsf{X}_P\vec{t}$ and for Skolem maps: $\mathsf{X}_P\vec{t} \Rightarrow_{\mathtt{Bool}} P(\vec{t}, (\Phi_P\vec{t}))$, for any predicate P of T.

We denote with $\bot$ the atomic formula $\mathtt{False}$ and will sometimes write a generic atomic formula as $P(t_1, \dots, t_n)$ rather than in the form $Pt_1 \dots t_n$. Finally, since any arithmetical formula has only variables of type $\mathtt{N}$, we shall freely omit their types, writing for instance $\forall x.A$ in place of $\forall x^{\mathtt{N}}.A$. Post rules cover many rules with atomic assumptions and conclusion as we find useful, for example, the rule: "if $f(z) \leq 0$ then $f(z) = 0$".

We defined $\Rightarrow_{\mathtt{Bool}}$: $\mathtt{Bool}, \mathtt{Bool} \to \mathtt{Bool}$ as a term implementing implication, therefore, to be accurate, the axiom $P(t_1, \dots, t_n, t) \Rightarrow_{\mathtt{Bool}} \mathsf{X}_P t_1 \dots t_n$ is not an implication between two atomic formulas, but it is equal to the single atomic formula $Qt_1 \dots t_n t$, where

$$Q = \lambda x_1^{\mathtt{N}} \dots \lambda x_{n+1}^{\mathtt{N}}. \Rightarrow_{\mathtt{Bool}} (Px_1 \dots x_n x_{n+1})(\mathsf{X}_P x_1 \dots x_{n+1})$$

Similarly, $\neg_{\mathtt{Bool}} P(\vec{t}, t)$ will denote a single atomic formula. Any atomic formula A of $\mathcal{L}_{\text{Class}}$ is a boolean term of $\mathcal{T}_{\text{Class}}$, therefore for any state constant s we may form the "finite approximation" $A[s] : \mathtt{Bool}, A[s] \in \mathcal{T}_{\text{Learn}}$ of A. In $A[s]$ we replace all oracles X_P and all Skolem maps Φ_P we have in A by their finite approximation $\chi_P s, \phi_P s$, computed with respect to the state constant s. We denote with $\mathcal{L}_{\text{Learn}}$ the set of all expressions $A[s]$ with $A \in \mathcal{L}_{\text{Class}}$ and s a state constant. All $A[s] \in \mathcal{L}_{\text{Learn}}$ may be interpreted by first order arithmetical formulas having all closed atomic subformulas decidable.

Using the metaphor explained in the introduction, we use a set of falsifiable hypotheses determined by s to predict a computable truth value $A[s] : \mathtt{Bool}$ in $\mathcal{T}_{\text{Learn}}$ for an atomic formula $A \in \mathcal{L}_{\text{Class}}$ that we cannot effectively evaluate. Our definition of realizability provides a formal semantics for the Extended Intuitionistic Arithmetic $\mathsf{HA}+\mathsf{EM}_1$, and therefore also for the more usual language of Arithmetic HA, in which all functions represent recursive maps.

DEFINITION 3.3.2 (TYPES FOR REALIZERS). For each arithmetical formula A we define a type $|A|$ of T by induction on A:

(1) $|P(t_1, \ldots, t_n)| = \mathtt{S}$,

(2) $|A \wedge B| = |A| \times |B|$,

(3) $|A \vee B| = \mathtt{Bool} \times (|A| \times |B|)$,

(4) $|A \to B| = |A| \to |B|$,

(5) $|\forall x A| = \mathtt{N} \to |A|$,

(6) $|\exists x A| = \mathtt{N} \times |A|$

We now define the realizability relation $t \Vdash A$, where $t \in \mathcal{T}_{\text{Class}}$, $A \in \mathcal{L}_{\text{Class}}$, t has state $\varnothing$ and $t : |A|$.

Definition 3.3.3 (Realizability). Assume s is a state constant, $t \in \mathcal{T}_{\text{Class}}$ is a closed term of state $\varnothing$, $C \in \mathcal{L}_{\text{Class}}$ is a closed formula, and $t : |C|$. Let $\vec{t} = t_1, \ldots, t_n : \mathtt{N}$. We define first the relation $t \Vdash_s C$ by induction and by cases according to the form of C:

(1) $t \Vdash_s P(\vec{t})$ if and only if $t[s] = \varnothing$ in $\mathcal{T}_{\text{Learn}}$ implies $P(\vec{t})[s] = \mathtt{True}$

(2) $t \Vdash_s A \wedge B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$

(3) $t \Vdash_s A \vee B$ if and only if either $p_0 t[s] = \mathtt{True}$ in $\mathcal{T}_{\text{Learn}}$ and $p_1 t \Vdash_s A$, or $p_0 t[s] = \mathtt{False}$ in $\mathcal{T}_{\text{Learn}}$ and $p_2 t \Vdash_s B$

(4) $t \Vdash_s A \to B$ if and only if for all u, if $u \Vdash_s A$, then $tu \Vdash_s B$

(5) $t \Vdash_s \forall x A$ if and only if for all numerals n, $tn \Vdash_s A[n/x]$

(6) $t \Vdash_s \exists x A$ if and only for some numeral n, $\pi_0 t[s] = n$ in $\mathcal{T}_{\text{Learn}}$ and $\pi_1 t \Vdash_s A[n/x]$

We define $t \Vdash A$ if and only if for all state constants s, $t \Vdash_s A$.

The definition of $\Vdash$ formalizes all the idea we sketched in the introduction. A realizer is a term t of $\mathcal{T}_{\text{Class}}$, possibly containing the non-computable functions X_P, Φ_P; if such functions were computable, t would be an intuitionistic realizer. Since in general t is not computable, we calculate its approximation $t[s]$ at state s, which is a term of $\mathcal{T}_{\text{Learn}}$, and we require it to satisfy the indexed-by-state realizability clauses. Realizers of disjunctions and existential statements provide a witness, which is an individual depending on an actual state of knowledge, representing all the hypotheses used to approximate the non-computable. The actual behavior of a realizer depends upon the current state of knowledge. The state is used only when there is relevant information about the truth of a given formula to be computed: the truth value $P(t_1, \ldots, t_n)[s]$ of an atomic formula and the disjunctive witness $p_0 t[s]$ and the existential witness $\pi_0 u[s]$ are computed w.r.t. the constant state s. A realizer t of $A \vee B$ uses s to predict which one between A and B is realizable (if $p_0 t[s] = \mathtt{True}$ then

A is realizable, and if $p_0 t[s] = \mathtt{False}$ then B is realizable). A realizer u of $\exists x A$ uses s to predict that $\pi_0 u[s]$ equals an n, some witness for $\exists x A$ (i.e. that $A[n/x]$ is realizable). These predictions need not be always correct; hence, it is possible that a realized atomic formula is actually false; we may have $t \Vdash_s P$ and $P[s] = \mathtt{False}$ in $\mathcal{T}_{\text{Learn}}$. If an atomic formula, although predicted to be true, is indeed false, then we have encountered a counterexample and so our theory is wrong, our approximation still inadequate; in this case, $t[s] \neq \varnothing$ by definition of $t \Vdash_s P$, and the atomic realizer t takes s and extends it to a larger state s', union of s and $t[s]$. That is to say: if something goes wrong, we must learn from our mistakes. The point is that after every learning, the actual state of knowledge grows, and if we ask to the same realizer new predictions, we will obtain "better" answers.

Indeed, we can say more about this last point. Suppose for instance that $t \Vdash A \vee B$ and let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. sequence. Then, since $t : \mathtt{Bool} \times |A| \times |B|$, then $p_0 t : \mathtt{Bool}$ is a closed term of $\mathcal{T}_{\text{Class}}$, converging in $\{s_i\}_{i\in\mathbb{N}}$ to a boolean; thus the sequence of predictions $\{p_0 t[s_i]\}_{i\in\mathbb{N}}$ eventually stabilizes, and hence a witness is eventually learned in the limit.

In the atomic case, in order to have $t \Vdash_s P(t_1, \dots, t_n)$, we require that if $t[s] = \varnothing$, then $P(t_1, \dots, t_n)[s] = \mathtt{True}$ in $\mathcal{T}_{\text{Learn}}$. That is to say: if t has no new information to add to s, then t must assure the truth of $P(t_1, \dots, t_n)$ w.r.t. s. By the zero theorem 3.2.15, when $t : \mathtt{S}$ is closed, there is plenty of state constants s such that $t[s] = \varnothing$; hence search for truth will be for us *computation of a zero*, driven by the excluded-middle instances and the Skolem axioms used by the proof, rather than exhaustive search for counterexamples. In chapter 5 we will prove that, actually, zeros for terms of $\mathcal{T}_{\text{Class}}$ can be computed by learning processes whose length can be bounded through constructive reasoning.

It is useful to give a slightly different definition of indexed realizability, which in some situations is *slightly* easier to reason with. The difference with definition 4.2.4 is only that the relation we are going to define now is between terms of $\mathcal{T}_{\text{Learn}}$ and formulas of $\mathcal{L}_{\text{Learn}}$, which are from the beginning approximations at some state s respectively of terms of $\mathcal{T}_{\text{Class}}$ and formulas of $\mathcal{L}_{\text{Class}}$.

Definition 3.3.4 (Variant of Indexed Realizability). Let s be a state constant. Assume $t \in \mathcal{T}_{\text{Learn}}$ and $A \in \mathcal{L}_{\text{Learn}}$ are of the form $t = t'[s], A = A'[s]$ for some closed $t' \in \mathcal{T}_{\text{Class}}$ of state $\varnothing$ and some closed $A' \in \mathcal{L}_{\text{Class}}$. We define $t \Vdash_s A$ for any state constant s by induction on A.

(1) $t \Vdash_s P(t_1, \dots, t_n)$ if and only if $t = \varnothing$ in $\mathcal{T}_{\text{Learn}}$ implies $P(t_1, \dots, t_n) = \mathtt{True}$

(2) $t \Vdash_s A \wedge B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$

(3) $t \Vdash_s A \vee B$ if and only if: either $p_0 t = \mathtt{True}$ in $\mathcal{T}_{\text{Learn}}$ and $p_1 t \Vdash_s A$, or $p_0 t = \mathtt{False}$ and $p_2 t \Vdash_s B$

(4) $t \Vdash_s A \to B$ if and only if for all u, if $u \Vdash_s A$, then $tu \Vdash_s B$

(5) $t \Vdash_s \forall x A$ if and only if for all numerals n, $tn \Vdash_s A[n/x]$

(6) $t \Vdash_s \exists x A$ if and only if for some numeral n $\pi_0 t = n$ in $\mathcal{T}_{\text{Learn}}$ and $\pi_1 t \Vdash_s A[n/x]$

The realizability relation is compatible with equality in $\mathcal{T}_{\text{Learn}}$:

LEMMA 3.3.5 . *If $t_1 \Vdash_s A[u_1/x]$, $t_1 = t_2$ and $u_1 = u_2$ in $\mathcal{T}_{Learn}$, then $t_2 \Vdash_s A[u_2/x]$*

PROOF. By straightforward induction on A.

□

We can now characterize $\Vdash\!\vdash$ in the following way.

LEMMA 3.3.6 (ALTERNATIVE CHARACTERIZATION OF REALIZABILITY). *Assume $t \in \mathcal{T}_{Class}$ is a closed term, $A \in \mathcal{L}_{Class}$ is a closed formula, and $t : |A|$. Then*

$$t \Vdash\!\vdash A \text{ if and only if for all state constants } s,\ t[s] \Vdash_s A[s]$$

PROOF. By definition unfolding and by induction on A, one shows that $t \Vdash\!\vdash_s A$ if and only if $t[s] \Vdash_s A[s]$.

□

EXAMPLE 3.3.7 . The most remarkable feature of our Realizability Semantics is the existence of a realizer E_P for EM_1. Assume that P is a predicate of T and define

$$\mathsf{E}_P := \lambda\vec{\alpha}^{\mathtt{N}}\langle \mathsf{X}_P\vec{\alpha},\ \langle \Phi_P\vec{\alpha},\ \varnothing\rangle,\ \lambda n^{\mathtt{N}}\ \mathsf{Add}_P\vec{\alpha}n\rangle$$

Indeed E_P realizes its associated instance of EM_1.

PROPOSITION 3.3.8 (REALIZER E_P OF EM_1).

$$\mathsf{E}_P \Vdash\!\vdash \forall\vec{x}.\ \exists y\ P(\vec{x}, y) \vee \forall y \neg_{\mathtt{Bool}} P(\vec{x}, y)$$

PROOF. Let $\vec{m}$ be a vector of numerals. $\mathsf{E}_P\vec{m}[s]$ is equal to

$$\langle \chi_P s\vec{m},\ \langle \varphi_P s\vec{m},\ \varnothing\rangle,\ \lambda n^{\mathtt{N}}\ \mathsf{add}_P s\vec{m}n\rangle$$

and we want to prove that

$$\mathsf{E}_P\vec{m}[s] \Vdash_s \exists y\ P(\vec{m}, y) \vee \forall y \neg_{\mathtt{Bool}} P(\vec{m}, y)$$

We have $p_0\mathsf{E}_P\vec{m}[s] = \chi_P s\vec{m}$ in $\mathcal{T}_{\text{Learn}}$. There are two cases.

(1) $\chi_P s\vec{m} = \mathtt{True}$. Then $\langle P, \vec{m}, n\rangle \in |s|$ for some numeral n such that $P(\vec{m}, n) = \mathtt{True}$, and we have to prove

$$p_1\mathsf{E}_P m[s] \Vdash_s \exists y\ P(\vec{m}, y)$$

By definition of $\varphi_P s\vec{m}$

$$p_1\mathsf{E}_P m[s] = \langle \varphi_P s\vec{m}, \varnothing\rangle = \langle n, \varnothing\rangle$$

Thus

$$\pi_0(p_1\mathsf{E}_P m)[s] = \pi_0\langle n, \varnothing\rangle = n$$

and

$$\pi_1(p_1\mathsf{E}_P m)[s] \Vdash_s P(\vec{m}, n)$$

because $P(\vec{m}, n) = \mathtt{True}$. We conclude

$$p_1\mathsf{E}_P m[s] \Vdash_s \exists y\ P(\vec{m}, y)$$

(2) $\chi_P s\vec{m} = \mathtt{False}$. Then $\langle P, \vec{m}, l\rangle \notin |s|$ for all numerals l. We have to prove

$$p_2 \mathsf{E}_P \vec{m}[s] = \lambda n\ \mathsf{add}_P s\vec{m}n \ \Vdash_s \forall y \neg_{\mathtt{Bool}} P(m, y)$$

i.e. that given any numeral n

$$\mathsf{add}_P s\vec{m}n \Vdash_s \neg_{\mathtt{Bool}} P(m, n)$$

By the definition of realizer in this case, we have to assume that $\mathsf{add}_P s\vec{m}n = \varnothing$, and prove that $\neg_{\mathtt{Bool}} P(\vec{m}, n)[s] = \mathtt{True}$. The substitution $(.)[s]$ has an empty effect over $P(\vec{m}, n)$, therefore we have to prove that $\neg_{\mathtt{Bool}} P(\vec{m}, n) = \mathtt{True}$, that is, that $P(\vec{m}, n) = \mathtt{False}$. Assume for contradiction that $P(\vec{m}, n) = \mathtt{True}$. We already proved that $\langle P, \vec{m}, l\rangle \notin |s|$, for all numerals l: from this and $P(\vec{m}, n) = \mathtt{True}$ we deduce that by definition $\mathsf{add}_P s\vec{m}n = |\{\langle P, \vec{m}, n\rangle\}|^{-1}$, contradiction.

□

E_P works according to the ideas we sketched in the introduction. It uses χ_P to make predictions about which one between $\exists y\ P(\vec{m}, y)$ and $\forall y \neg_{\mathtt{Bool}} P(\vec{m}, y)$ is true. χ_P, in turn, relies on the constant s denoting the actual state to make its own prediction. If $\chi_P sm = \mathtt{False}$, given any n, $\neg_{\mathtt{Bool}} P(m, n)$ is predicted to be true; if it is not the case, we have a counterexample and add_P requires to extend the state with $\langle P, \vec{m}, n\rangle$. On the contrary, if $\chi_P sm = \mathtt{True}$, there is unquestionable evidence that $\exists y P(\vec{m}, y)$ holds; namely, there is some numeral n such that $\langle P, \vec{m}, n\rangle$ is in s; then φ_P is called, and it returns $\varphi_P s\vec{m} = n$.

This is the basic mechanism by which we implement learning: every state extension is linked with an assumption about an instance of EM_1 which we used and turned out to be wrong (this is the only way to come across a counterexample); in next computations, the actual state will be bigger, the realizer will not do the same error, and hence will be "wiser".

As usual for a realizability interpretation, we may extract from any realizer $t \Vdash \forall x.\exists y.P(x, y)$, with $P \in \mathsf{T}$, some recursive map f from the set of numerals to the set of numerals, such that $P(n, f(n))$ for all numerals n.

Example 3.3.9 (Program Extraction via Learning Based Realizability). Let t be a term of $\mathcal{T}_{\text{Class}}$ and suppose that $t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$, with P atomic. Then, from t one can effectively define a recursive function f from the set of numerals to the set of numerals such that for every numeral n, $Pn(f(n)) = \mathtt{True}$.

Proof. Let

$$v := \lambda m^{\mathtt{N}}\ \pi_1(tm)$$

v is of type $\mathtt{N} \to \mathtt{S}$. By zero theorem 3.2.15, there exists a recursive function zero from the set of numerals to the set of state constants such that $vn[\mathsf{zero}(n)] = \varnothing$ for every numeral n. Define f as the function

$$m \mapsto \pi_0(tm)[\mathsf{zero}(m)]$$

and fix a numeral n. By unfolding the definition of realizability with respect to the state $\mathsf{zero}(n)$, we have that

$$tn \Vdash_{\mathsf{zero}(n)} \exists y^{\mathtt{N}} Pny$$

and hence

$$\pi_1(tn) \Vdash_{\mathsf{zero}(n)} Pn(f(n))$$

that is to say

$$vn[\mathsf{zero}(n)] = \varnothing \implies Pn(f(n)) = \mathtt{True}$$

and therefore

$$Pn(f(n)) = \mathtt{True}$$

which is the thesis.

□

REMARK 3.3.10 . In chapter 5 we shall prove that the map f constructed in example 3.3.9 is even definable in Gödel's T. This result formally proves that our realizability interpretation is a constructive semantics and that f is not a brute force search algorithm. More precisely, we can argue as follows. The numeral $f(n)$ is computed by finding a zero of $vn = \pi_1(tn)$, i.e. a state s such that $\pi_1(tn)[s] = \varnothing$. By the Zero theorem 3.2.15, this zero is computed step by step by constructing the sequence $s_0 = \varnothing$, $s_{n+1} = s_n \Cup t[s_n]$ and stopping at the first m such that $\pi_1(tn)[s_m] = \varnothing$.
First, we observe that each portion of s_m is efficiently constructed: for each n, the state s_n is efficiently extended to s_{n+1}, through the addition of new oracle values learned by t by counterexamples, i.e. by the falsification of some excluded middle or Skolem axiom instances. No brute force search whatsoever, thus, for new oracle values: they are all efficiently produced by $t[s_n]$, which, modulo some trivial coding, *is* a term of Gödel's T and just cannot search blindly for oracle values, since it is a primitive recursive functional of finite type.
Secondly, an upper bound to m can be computed in Gödel's T, as proven in chapter 5, theorem 5.6.2. Moreover, this upper bound results from a constructive proof of the Zero theorem. Hence, s_m and thus $f(n)$ can be defined by a primitive recursive functional of finite type, which, again, by construction cannot explore blindly the infinite search space of the knowledge states in order to find a zero.

Moreover, from the low level computational point of view and in the language of ϵ-substitution method, our realizers represent convergent procedures to find out a "solving substitution", i.e. a state representing an approximation of Skolem functions (i.e., ϵ-terms) which makes true the Skolem axioms instances used in a proof of an existential statement. The advantage of our semantics is the possibility of defining such procedures directly from high level proofs, by means of Curry-Howard correspondence, hence avoiding the round-about route which forces to use a quantifier free deduction system. In the case of a provable formula in the language of Peano Arithmetic (that is, one not containing the symbols X_P or Φ_P) we do not need at all to modify the language of its proof and to use the Skolem axioms χ, φ.

Now we explain how to turn each proof $\mathcal{D}$ of a formula $A \in \mathcal{L}_{\text{Class}}$ in $\mathsf{HA} + \mathsf{EM}_1$ into a realizers $\mathcal{D}^*$ of the same A. By induction on $\mathcal{D}$, we define a "decoration with realizers" $\mathcal{D}^{\text{Real}}$ of $\mathcal{D}$, in which each formula B of $\mathcal{D}$ is replaced by a new statement $u \vdash B$, for some $u \in \mathcal{T}_{\text{Class}}$ of state $\varnothing$. If $t \vdash A$ is the conclusion of $\mathcal{D}^{\text{Real}}$, we set $\mathcal{D}^* = t$. Then we will prove that if $\mathcal{D}$ is closed and without assumptions, then $\mathcal{D}^* \in \mathcal{T}_{\text{Class}}$ and $\mathcal{D}^* \Vdash A$. The decoration $\mathcal{D}^{\text{Real}}$ of $\mathcal{D}$ with realizers is completely standard: we have new realizers only for EM_1 and for atomic formulas. For notation simplicity, if x_i is the label for the set of occurrences of some assumption A_i of $\mathcal{D}$, we use x_i also as a name of one free variable in $\mathcal{D}^*$ of type $|A_i|$.

If T is any type of $\mathcal{T}_{\mathtt{S}}$, we denote with d^T a dummy term of type T, defined by $d^{\mathtt{N}} = 0$, $d^{\mathtt{Bool}} = \mathtt{False}$, $d^{\mathtt{S}} = \varnothing$, $d^{A\to B} = \lambda\_^A.d^B$ (with $\_^A$ any variable of type A), $d^{A\times B} = \langle d^A, d^B\rangle$.

Definition 3.3.11 (Term Assignment Rules for $\mathsf{HA}+\mathsf{EM}_1$). Assume $\mathcal{D}$ is a proof of $A \in \mathcal{L}_{\text{Class}}$ in $\mathsf{HA}+\mathsf{EM}_1$, with free assumptions $A_1,\dots,A_n$ denoted by proof variables $x_1^{A_1},\dots,x_n^{A_n}$ and free integer variables $\alpha_1^{\mathtt{N}},\dots,\alpha_m^{\mathtt{N}}$. By induction on $\mathcal{D}$, we define a decorated proof-tree $\mathcal{D}^{\text{Real}}$, in which each formula B is replaced by $u \vdash B$ for some $u \in \mathcal{T}_{\text{Class}}$, and the conclusion A with some $t \vdash A$, with $FV(t) \subseteq \{x_1^{|A_1|},\dots,x_1^{|A_1|},\alpha_1^{\mathtt{N}},\dots,\alpha_m^{\mathtt{N}}\}$. Eventually we set $\mathcal{D}^* = t$.

(1) $\dfrac{}{x^{|A|} \vdash A}$ if $\mathcal{D}$ consists of a single free assumption $A \in \mathcal{L}_{\text{Class}}$ labeled x^A.

(2) $\dfrac{u \vdash A \quad t \vdash B}{\langle u,t\rangle \vdash A\wedge B}$ $\qquad \dfrac{u \vdash A\wedge B}{\pi_0 u \vdash A}$ $\qquad \dfrac{u \vdash A\wedge B}{\pi_1 u \vdash B}$

(3) $\dfrac{u \vdash A\to B \quad t \vdash A}{ut \vdash B}$ $\qquad \dfrac{u \vdash B}{\lambda x^{|A|}u \vdash A\to B}$

(4) $\dfrac{u \vdash A}{\langle \mathtt{True}, u, d^B\rangle \vdash A\vee B}$ $\qquad \dfrac{u \vdash B}{\langle \mathtt{False}, d^A, u\rangle \vdash A\vee B}$

$$\frac{u \vdash A\vee B \quad w_1 \vdash C \quad w_2 \vdash C}{\mathsf{if}\ p_0 u\ \mathsf{then}\ (\lambda x^{|A|}w_1)(p_1 u)\ \mathsf{else}\ (\lambda x^{|B|}w_2)(p_2 u) \vdash C}$$

where d^A and d^B are dummy closed terms of $\mathcal{T}_{\text{Class}}$ of type $|A|$ and $|B|$

(5) $\dfrac{u \vdash \forall\alpha A}{ut \vdash A[t/\alpha]}$ $\dfrac{u \vdash A}{\lambda\alpha^{\mathtt{N}}u \vdash \forall\alpha A}$

where t is a term of $\mathcal{L}_{\text{Class}}$ and $\alpha^{\mathtt{N}}$ does not occur free in any free assumption B of the subproof of $\mathcal{D}$ of conclusion A.

(6) $\dfrac{u \vdash A[t/\alpha^{\mathtt{N}}]}{\langle t,u\rangle \vdash \exists\alpha^{\mathtt{N}}.A}$ $\qquad \dfrac{u \vdash \exists\alpha^{\mathtt{N}}.A \quad t \vdash C}{(\lambda\alpha^{\mathtt{N}}\lambda x^{|A|}\ t)(\pi_0 u)(\pi_1 u) \vdash C}$

where $\alpha^{\mathtt{N}}$ is not free in C nor in any free assumption B different from A in the subproof of $\mathcal{D}$ of conclusion C.

(7) $$\frac{u \vdash A(0) \quad v \vdash \forall\alpha.A(\alpha)\to A(\mathsf{S}(\alpha))}{\lambda\alpha^{\mathtt{N}}\mathsf{R}uv\alpha \vdash \forall\alpha A}$$

(8) $$\frac{u_1 \vdash A_1\ u_2 \vdash A_2\ \cdots\ u_n \vdash A_n}{u_1 \Cup u_2 \Cup \cdots \Cup u_n \vdash A}$$

where $n > 0$ and $A_1, A_2, \dots, A_n, A$ are atomic formulas of $\mathcal{L}_{\text{Class}}$, and the rule is a Post rule for equality or ordering, or a tautological consequence.

(9) $\dfrac{}{\varnothing \vdash A}$

where A is an atomic axiom of $\mathsf{HA}+\mathsf{EM}_1$ (an axiom of equality or of ordering or a tautology or an equation of T)

(10) $\dfrac{}{\mathsf{E}_P \vdash \forall \vec{x}.\ \exists y\ P(\vec{x},y) \vee \forall y \neg_{\mathtt{Bool}} P(\vec{x},y)}$
where P is a predicate of T.

(11) $\dfrac{}{\mathsf{Add}_P \vec{t}, t \vdash P(\vec{t},t) \Rightarrow_{\mathtt{Bool}} \mathsf{X}_P \vec{t}}$, χ-Axiom

(12) $\dfrac{}{\varnothing \vdash \mathsf{X}_P \vec{t} \Rightarrow_{\mathtt{Bool}} P(\vec{t}, (\Phi_P \vec{t}))}$, φ-Axiom

The term decorating the conclusion of a Post rule is of the form $u_1 \Cup \cdots \Cup u_n$. In this case, we have n different realizers, whose learning capabilities are put together through a sort of union. By Lemma 3.2.2.2, if $u_1 \Cup \cdots \Cup u_n[s] = \varnothing$, then $u_1[s] = \ldots = u_n[s] = \varnothing$, i.e. all u_i "have nothing to learn". In that case, each u_i must guarantee A_i to be true, and therefore the conclusion of the Post rule is true, because true premises $A_1, \ldots, A_n$ spell a true conclusion A.

We now prove our main theorem, that every theorem of $\mathsf{HA} + \mathsf{EM}_1$ is realizable.

THEOREM 3.3.12 (ADEQUACY THEOREM). *Suppose that $\mathcal{D}$ is a proof of A in the system $\mathsf{HA} + \mathsf{EM}_1$ with free assumptions $x_1^{A_1}, \ldots, x_n^{A_n}$ and free variables $\alpha_1 : \mathtt{N}, \ldots, \alpha_k : \mathtt{N}$. Let $w = \mathcal{D}^*$. For all state constants s and for all numerals $n_1, \ldots, n_k$, if*

$$t_1[s] \Vdash_s A_1[n_1/\alpha_1 \cdots n_k/\alpha_k][s], \ldots, t_n[s] \Vdash_s A_n[n_1/\alpha_1 \cdots n_k/\alpha_k][s]$$

then

$$w[t_1/x_1^{|A_1|} \cdots t_n/x_n^{|A_n|}\ n_1/\alpha_1 \cdots n_k/\alpha_k][s] \Vdash_s A[n_1/\alpha_1 \cdots n_k/\alpha_k][s]$$

PROOF. Notation: for any term v and formula B, we denote

$$v[t_1/x_1^{|A_1|} \cdots t_n/x_n^{|A_n|}\ n_1/\alpha_1 \cdots n_k/\alpha_k][s]$$

with $\overline{v}$ and $B[n_1/\alpha_1 \cdots n_k/\alpha_k][s]$ with $\overline{B}$. We have $|\overline{B}| = |B|$ for all formulas B. We denote with $=$ the provable equality in $\mathcal{T}_{\text{Learn}}$. We proceed by induction on w. Consider the last rule in the derivation $\mathcal{D}$:

(1) If it is the rule for variables, then $w = x_i^{|A_i|} = x^{|\overline{A_i}|}$ and $A = A_i$. So $\overline{w} = t_i \Vdash_s \overline{A_i} = \overline{A}$.

(2) If it is the $\wedge I$ rule, then $w = \langle u, t\rangle$, $A = B \wedge C$, $u \vdash B$ and $t \vdash C$. Therefore, $\overline{w} = \langle \overline{u}, \overline{t}\rangle$. By induction hypothesis, $\pi_0 \overline{w} = \overline{u} \Vdash_s \overline{B}$ and $\pi_1 \overline{w} = \overline{t} \Vdash_s \overline{C}$; so, by definition, $\overline{w} \Vdash_s \overline{B} \wedge \overline{C} = \overline{A}$.

(3) If it is a $\wedge E$ rule, say left, then $w = \pi_0 u$ and $u \vdash A \wedge B$. So $\overline{w} = \pi_0 \overline{u} \Vdash_s \overline{A}$, because $\overline{u} \Vdash_s \overline{A} \wedge \overline{B}$ by induction hypothesis.

(4) If it is the $\to E$ rule, then $w = ut$, $u \vdash B \to A$ and $t \vdash B$. So $\overline{w} = \overline{u}\overline{t} \Vdash_s \overline{A}$, for $\overline{u} \Vdash_s \overline{B} \to \overline{A}$ and $\overline{t} \Vdash_s \overline{B}$ by induction hypothesis.

(5) If it is the $\to I$ rule, then $w = \lambda x^{|B|} u$, $A = B \to C$ and $u \vdash C$. Thus, $\overline{w} = \lambda x^{|B|} \overline{u}$. Suppose now that $t \Vdash_s \overline{B}$; by induction hypothesis on u, $\overline{w} t = \overline{u}[t/x^{|B|}] \Vdash_s \overline{C}$.

(6) If it is a $\vee I$ rule, say left, then $w = \langle \mathtt{True}, u, d^C \rangle$, $A = B \vee C$ and $u \vdash B$. So, $\overline{w} = \langle \mathtt{True}, \overline{u}, d^C \rangle$ and hence $p_0\overline{w} = \mathtt{True}$. We indeed verify that $p_1\overline{w} = \overline{u} \Vdash_s \overline{B}$ with the help of induction hypothesis.

(7) If it is a $\vee E$ rule, then

$$w = \mathsf{if}\ p_0 u\ \mathsf{then}\ (\lambda x^{|B|} w_1) p_1 u\ \mathsf{else}\ (\lambda y^{|C|} w_2) p_2 u$$

and $u \vdash B \vee C, w_1 \vdash D, w_2 \vdash D, A = D$. So,

$$\overline{w} = \mathsf{if}\ p_0 \overline{u}\ \mathsf{then}\ (\lambda x^{|B|} \overline{w_1}) p_1 \overline{u}\ \mathsf{else}\ (\lambda y^{|C|} \overline{w_2}) p_2 \overline{u}$$

Assume $p_0\overline{u} = \mathtt{True}$. Then by inductive hypothesis $p_1\overline{u} \Vdash_s \overline{B}$, and again by induction hypothesis, $\overline{w} = \overline{w}_1[p_1\overline{u}/x^{|\overline{B}|}] \Vdash_s \overline{D}$. Symmetrically, if $p_0\overline{u} = \mathtt{False}$, then $\overline{w} \Vdash_s \overline{D}$.

(8) If it is the $\forall E$ rule, then $w = ut$, $A = B[t/\alpha]$ and $u \vdash \forall \alpha B$. So, $\overline{w} = \overline{u}\overline{t}$. For some numeral n we have $n = \overline{t}$. By inductive hypothesis $\overline{u} \Vdash_s \forall \alpha \overline{B}$, therefore $\overline{u}\overline{t} = \overline{u}n \Vdash_s \overline{B}[n/\alpha] = \overline{B}[\overline{t}/\alpha] = \overline{A}$.

(9) If it is the $\forall I$ rule, then $w = \lambda \alpha^{\mathtt{N}} u$, $A = \forall \alpha B$ and $u \vdash B$. So, $\overline{w} = \lambda \alpha^{\mathtt{N}} \overline{u}$. Let n be a numeral; we have to prove that $\overline{w}n = \overline{u}[n/\alpha] \Vdash_s \overline{B}[n/\alpha]$, which is true, indeed, by induction hypothesis.

(10) If it is the $\exists E$ rule, then $w = (\lambda \alpha^{\mathtt{N}} \lambda x^{|B|} t)(\pi_0 u)(\pi_1 u)$, $t \vdash A$ and $u \vdash \exists \alpha^{\mathtt{N}}.B$. Assume $n = \pi_0 u$, for some numeral n. Then

$$\overline{t}[n/\alpha^{\mathtt{N}}, \pi_1\overline{u}/x^{|\overline{B}[n/\alpha^{\mathtt{N}}]|}] \Vdash_s \overline{A}[n/\alpha] = \overline{A}$$

by inductive hypothesis, whose application being justified by the fact, also by induction, that $\overline{u} \Vdash_s \exists \alpha^{\mathtt{N}}.\overline{B}$ and hence $\pi_1\overline{u} \Vdash_s \overline{B}[n/\alpha^{\mathtt{N}}]$. We thus obtain

$$\overline{w} = \overline{t}[\pi_0\overline{u}/\alpha^{\mathtt{N}}\ \pi_1\overline{u}/x^{|B|}] \Vdash_s \overline{A}[n/\alpha] = \overline{A}$$

(11) If it is the $\exists I$ rule, then $w = \langle t, u \rangle$, $A = \exists \alpha B$, $u \vdash B[t/\alpha]$. So, $\overline{w} = \langle \overline{t}, \overline{u} \rangle$; and, indeed, $\pi_1\overline{w} = \overline{u} \Vdash_s \overline{B}[\pi_0\overline{w}/\alpha] = \overline{B}[\overline{t}/\alpha]$ since by induction hypothesis $\overline{u} \Vdash_s \overline{B}[\overline{t}/\alpha]$.

(12) If it is the induction rule, then $w = \lambda \alpha^{\mathtt{N}}\ \mathsf{R}uv\alpha$, $A = \forall \alpha B$, $u \vdash B(0)$ and $v \vdash \forall \alpha.B(\alpha) \to B(\mathsf{S}(\alpha))$. So, $\overline{w} = \lambda \alpha^{\mathtt{N}} \mathsf{R}\overline{u}\overline{v}\alpha$. Now let n be a numeral. A plain induction on n shows that $\overline{w}n = \mathsf{R}\overline{u}\overline{v}n \Vdash_s \overline{B}[n/\alpha]$, for $\overline{u} \Vdash_s \overline{B}(0)$ and $\overline{v}i \Vdash_s \overline{B}(i) \to \overline{B}(\mathsf{S}(i))$ for all numerals i by induction hypothesis.

(13) If it is a Post rule, then $w = u_1 \Cup u_2 \Cup \cdots \Cup u_n$ and $u_i \vdash A_i$. So, $\overline{w} = \overline{u}_1 \Cup \overline{u}_2 \Cup \cdots \Cup \overline{u}_n$. Suppose now that $\overline{w}[s] = \varnothing$; then we have to prove that $\overline{A} = \mathtt{True}$. It suffices to prove that $\overline{A}_1 = \overline{A}_2 = \cdots = \overline{A}_n = \mathtt{True}$. By Lemma 3.2.2 we have $\overline{u}_1 = \cdots = \overline{u}_n = \varnothing$ and by induction hypothesis $\overline{A}_1 = \cdots = \overline{A}_n = \mathtt{True}$, since $\overline{u}_i \Vdash_s \overline{A}_i$, for $i = 1, \ldots, n$.

(14) If it is a χ-axiom rule, then $w = \mathsf{Add}_P t_1 \ldots t_n t$ and

$$A = P(t_1, \ldots, t_n, t) \Rightarrow \mathsf{X}_P t_1 \ldots t_n$$

Let $\vec{t} = \bar{t}_1, \ldots, \bar{t}_n$. For some numeral m we have $m = \bar{t}$. Suppose by contradiction that $\overline{w} = \varnothing$ and $P(\vec{t}, \bar{t}) = P(\vec{t}, m) = \mathtt{True}$ and $\chi_P s\vec{t} = \mathtt{False}$. From $\chi_P s\vec{t} = \mathtt{False}$ we get $\langle P, \vec{t}, m' \rangle \notin s$ for all numerals m'. We deduce $\overline{w} = \mathtt{add}_P s\vec{t}m = |\{\langle P, \vec{t}, m\rangle\}|^{-1}$, contradiction.

(15) w realizes an EM_1 axiom: this is Proposition 3.3.8.

(16) If it is a φ-axiom rule, then $w = \varnothing$ and

$$A = \mathsf{X}_P t_1 \ldots t_n \Rightarrow P(t_1, \ldots, t_n, (\Phi_P t_1 \ldots t_n))$$

We have $\overline{w} = \varnothing$. Let us denote $\vec{t} = \bar{t}_1 \ldots \bar{t}_n$. Suppose that $\chi_P s\vec{t} = \mathtt{True}$. Then for some numeral m we have $\langle P, \vec{t}, m\rangle \in s$ and $P\vec{t}m = \mathtt{True}$ and $\varphi_P s\vec{t} = m$. By definition of φ_P we have

$$P(\vec{t}, (\varphi_P s\vec{t})) = \mathtt{True}$$

We conclude that $\overline{A} = \mathtt{True}$.

□

COROLLARY 3.3.13 . *If A is a closed formula provable in $\mathsf{HA} + \mathsf{EM}_1$, then there exists $t \in \mathcal{T}_{Class}$ such that $t \Vdash A$.*

3.4. Conclusion and further works

Many notions of realizability for Classical Logic already exists. A notion similar to our one in spirit and motivations is Goodman's notion of Relative realizability [24]. However, there is an intrinsic difference between our solution and Goodman's solution. Goodman uses forcing to obtain a "static" description of learning. His "possible worlds" are learning states, but *there is no explicit operation updating a world to a larger word.* The dynamic aspect of learning (which is represented by a winning strategy in Game Semantics) is therefore lost. Using our realizability model, a realizer of an atomic formula, instead of being a trivial map, is *a map extending worlds*, whose fixed points are the worlds in which the atomic formula is true. Extending a world represents, in our realizability Semantics, the idea of "learning by trial-and-error" that we have in game semantics, while fixed points represent the final state of the game.

A second notion related to our realizability Semantics is Avigad's idea of "update procedure" [5]. A state s in our chapter corresponds to a finite model of skolem maps in Avigad. An "update procedure" is a construction "steering" the future evolution of a finite partial model s of skolem maps, to which our individuals belong, in a wanted direction. The main difference with our work is that we express this idea formally, by interpreting an "update procedure" as a realizer (in the sense of Kreisel) for a Skolem axiom. Another important difference is that our realizability relation is defined for all first-order formulas with Skolem maps, while the theory of "update procedures" is defined only for quantifier-free formulas with Skolem maps.

Another difference with the other realizability or Kripke models for Classical Logic is in the notion of individual and in the equality between individuals. Assume that m is the output of a skolem map for $\exists y.P(n, y)$, with P decidable, and $m = \{m[s]|s \in \mathtt{S}\}$ a family of values depending on the finite partial model s. Then our realizer for Skolem axioms "steers"

the evolution of s towards some universe in which the axiom $\exists y.P(n,y) \Rightarrow P(n,m[s])$ is true. Modifying the evolution of s may modify the value of $m[s]$. In our realizability Semantics we introduce a notion of individuality which is "dynamical" (depending on a state s) and "interactive" (the value of the individual depends on what a realizer does). This second aspect is new. A realizer may "try" to equate an individual $a = \{a[s]|s \in \mathtt{S}\}$ with another individual $b = \{b[s]|s \in \mathtt{S}\}$. Whenever this is possible, the realizer defines a construction over the evolution of the universe s producing such an effect, while a random evolution of s (without an "interaction" with the realizer) does not guarantee that eventually we have $a[s] = b[s]$. This is why, in our realizability model, even equality among concrete objects is not a "statical" fact, but it is the effect of applying a realizer (which is a construction over the evolution of the state or "world" s). In the other models either equality is "static", or, even when it is "dynamical", and it changes with time, it is not "interactive": the final truth value of an equality is not the effect of the application of the realizer, but it is eventually the same in all future evolutions of the current world.

Many aspects of this chapter will require some further work. A challenging idea is to iterate the construction we had for EM_1, in order to provide a learning model for the entire classical Arithmetic. In this case the leading concepts would be the game-theoretical notion of *"level of backtracking"*, introduced in [9] and [11], a notion related to the more informal notion of *non-monotonic learning*.

Another aspect deserving further work is comparing the programs extracted from classical proofs with our method and with other methods, say, with Friedman A-translation. Our interpretation, explaining in term of learning how the extracted program work, should allow us to modify and improve the extracted program in a way impossible for the more formal (but very elegant) A-translation.

We remarked that our interpretation is implicitly parametric with respect to the operation $\mathcal{U}$ merging the realizers of two atomic formulas. As explained in [12], by choosing different variant of this operation we may study different evaluation strategies for the extracted programs: sequential and parallel, left-to-right and right-to-left, confluent and non-confluent. We would like to study whether by choosing a particular evaluation strategy we may extract a more efficient program.

CHAPTER 4

Learning Based Realizability and 1-Backtracking Games

ABSTRACT. We prove a soundness and completeness result for learning based realizability with respect to 1-Backtracking Coquand game semantics. First, we prove that interactive learning based classical realizability is sound with respect to Coquand game semantics. In particular, any realizer of an implication-and-negation-free arithmetical formula embodies a winning recursive strategy for the 1-Backtracking version of Tarski games. We also give examples of realizer and winning strategy extraction for some classical proofs. Secondly, we extend our notion of realizability to a total recursive learning based realizability and show that the notion is complete with respect to Coquand semantics, when it is restricted to 1-Backtracking games.

4.1. Introduction

In this chapter we show that learning based realizability (see chapter 3) relates to 1-Backtracking Tarski games as intuitionistic realizability (see Kleene [31]) relates to Tarski games, when one considers implication-and-negation-free formulas. The relationship we refer to is between realizability on one hand, and existence of winning strategies on the other. In particular, it is known that a negation-and-implication-free arithmetical formula is Kleene realizable if and only if Eloise has a recursive winning strategy in the associated Tarski game. We show as well that an implication-and-negation-free arithmetical formula is "learning realizable" if and only if Eloise has recursive winning strategy in the associated 1-Backtracking Tarski game.

It is well known that Tarski games (which were actually introduced by Hintikka, see [29] and definition 4.3.3) are just a simple way of rephrasing the concept of classical truth in terms of a game between two players - the first one, Eloise, trying to show the truth of a formula, the second, Abelard, its falsehood - and that a Kleene realizer gives a recursive winning strategy to the first player. The result is quite expected: since a realizer gives a way of computing all the information about the truth of a formula, the player trying to prove the truth of that formula has a recursive winning strategy. However, not at all *any* classically provable arithmetical formula allows a winning recursive strategy for that player; otherwise, the decidability of the Halting problem would follow.

In [14], Coquand introduced a new game semantics for Peano Arithmetic, centered on the concept of "Backtracking Tarski game": a special Tarski game in which players have the additional possibility of correcting their moves and backtracking to a previous position of the game anytime they wish. Coquand then showed that for any provable negation-and-implication-free arithmetical formula A, Eloise has a *recursive* winning strategy in the Backtracking Tarski game associated to A. Remarkably, a proof in Peano Arithmetic thus hides a non trivial computational content that can be described as a recursive strategy that produces witnesses in classical Arithmetic by interaction and learning.

In the first part of this chapter, we show that learning based realizers have direct interpretation as recursive winning strategies in 1-Backtracking Tarski games (which are a particular case of Coquand games: see Berardi et al [9] and definition 4.3.2 below). The result was wished, because interactive learning based realizers, by design, are similar to strategies in games with backtracking: they improve their computational ability by learning from interaction and counterexamples in a convergent way; eventually, they gather enough information about the truth of a formula to win its associated game.

An interesting but incomplete step towards our result was the Hayashi realizability [27]. Indeed, a realizer in the sense of Hayashi represents a recursive winning strategy in 1-Backtracking games. However, from the computational point of view, Hayashi realizers do not relate to 1-Backtracking games in a significant way: Hayashi winning strategies work by exhaustive search and, actually, do not learn from the game and from the *interaction* with the other player. As a result of this issue, constructive upper bounds on the length of games cannot be obtained, whereas using our realizability it is possible. For example, in the case of the 1-Backtracking Tarski game for the formula $\exists x \forall y f(x) \leq f(y)$, the Hayashi realizer checks all the natural numbers to be sure that an n such that $\forall y f(n) \leq f(y)$ is eventually found. On the contrary, our realizer yields a strategy for Eloise which bounds the number of backtrackings by $f(0)$, as shown in this paper; moreover, what the strategy learns is uniquely determined by interaction with the other player. In this case, the Hayashi strategy is the same one suggested by the classical *truth* of the formula, whereas ours is the constructive strategy suggested by its classical *proof*.

Since learning based realizers are extracted from proofs in $\mathsf{HA}{+}\mathsf{EM}_1$ (Heyting Arithmetic with excluded middle over existential sentences, see chapter 3), one also has an interpretation of classical proofs as strategies with 1-Backtracking. Moreover, studying learning based realizers in terms of 1-Backtracking games also sheds light on their behaviour and offers an interesting case study in program extraction and interpretation in classical arithmetic.

In the second part of the chapter, we extend the class of learning based realizers from a classical version of Gödel's system T to a classical version of $\mathcal{PCF}$ and define a more general "total recursive learning based realizability". This step is analogous to the (conceptual, rather than chronological) step leading from Kreisel realizability to Kleene realizability: one extends the computational power of realizers. We then prove a completeness theorem: for every implication-and-negation-free arithmetical formula A, if Eloise has recursive winning strategy in the 1-Backtracking Tarski game associated to A, then A is also realizable.

The *plan of the chapter* is the following. In section §4.2, we recall the definitions and results from chapter 3 that we shall need in the present one. In section §4.3, we prove our first main theorem: a realizer of an arithmetical formula embodies a winning strategy in its associated 1-Backtracking Tarski game. In section §4.4, we extract realizers from two classical proofs and study their behavior as learning strategies. In section §4.5, we define an extension of the learning based realizability of chapter 3 and in section §4.6 prove its completeness with respect to 1-Backtracking Tarski games.

4.2. Learning-Based Realizability for the Standard Language of Arithmetic

In this chapter, we will use a standard language of Arithmetic: the symbols X_P, Φ_P will not occur in the language of formulas, but only in realizers. We recall the definition and results we need here.

DEFINITION 4.2.1 (CONVERGENCE). Assume that $\{s_i\}_{i\in\mathbb{N}}$ is a w.i. sequence of state constants, and $u, v \in \mathcal{T}_{\text{Class}}$.

(1) u converges in $\{s_i\}_{i\in\mathbb{N}}$ if $\exists i \in \mathbb{N}.\forall j \geq i.u[s_j] = u[s_i]$ in $\mathcal{T}_{\text{Learn}}$.

(2) u converges if u converges in every w.i. sequence of state constants.

We will make use of the following two theorems of chapter 3.

THEOREM 4.2.1 (STABILITY THEOREM). *Assume $t \in \mathcal{T}_{Class}$ is a closed term of atomic type A ($A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{S}\}$). Then t is convergent.*

THEOREM 4.2.2 (ZERO THEOREM). *Let $t : \mathtt{S}$ be a closed term of $\mathcal{T}_{Class}$ of state $\varnothing$ and s any state constant. Define, by induction on n, a sequence $\{s_n\}_{n\in\mathbb{N}}$ of state constants such that: $s_0 = s$ and $s_{n+1} = s_n \Cup t[s_n]$. Then, there exists an n such that $t[s_n] = \varnothing$.*

We now define a language for Peano Arithmetic and then formulate a realizability relation between terms of $\mathcal{T}_{\text{Class}}$ and formulas of the language.

DEFINITION 4.2.2 (THE LANGUAGE $\mathcal{L}$ OF PEANO ARITHMETIC). We define:

(1) The terms of $\mathcal{L}$ are all terms t of Gödel's system T, such that $t : \mathtt{N}$ and $FV(t) \subseteq \{x_1^{\mathtt{N}}, \ldots, x_n^{\mathtt{N}}\}$ for some $x_1, \ldots, x_n$.

(2) The atomic formulas of $\mathcal{L}$ are all terms $Qt_1 \ldots t_n$ of Gödel's system T, for some $Q : \mathtt{N}^n \to \mathtt{Bool}$ *closed term of* T, and some terms $t_1, \ldots, t_n$ of $\mathcal{L}$.

(3) The formulas of $\mathcal{L}$ are built from atomic formulas of $\mathcal{L}$ by the connectives $\vee, \wedge, \to$ $\forall, \exists$ as usual.

We now define the types realizers as in chapter 3 (we only use a different notation in order to avoid confusion in the rest of the chapter).

DEFINITION 4.2.3 (TYPES FOR REALIZERS). For each arithmetical formula A we define a type $[A]$ of T by induction on A: $[P(t_1, \ldots, t_n)] = \mathtt{S}$, $[A \wedge B] = [A] \times [B]$, $[A \vee B] = \mathtt{Bool} \times ([A] \times [B])$, $[A \to B] = [A] \to [B]$, $[\forall x A] = \mathtt{N} \to [A]$, $[\exists x A] = \mathtt{N} \times [A]$

We give the simplified notion of learning-based realizability we shall use in the following.

DEFINITION 4.2.4 (LEARNING-BASED REALIZABILITY). Assume s is a state constant, $t \in \mathcal{T}_{\text{Class}}$ is a closed term of state $\varnothing$, $A \in \mathcal{L}$ is a closed formula, and $t : [A]$. Let $\vec{t} = t_1, \ldots, t_n : \mathtt{N}$.

(1) $t \Vdash_s P(\vec{t})$ if and only if $t[s] = \varnothing$ in $\mathcal{T}_{\text{Learn}}$ implies $P(\vec{t}) = \mathtt{True}$

(2) $t \Vdash_s A \wedge B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$

(3) $t \Vdash_s A \vee B$ if and only if either $p_0 t[s] = \texttt{True}$ in $\mathcal{T}_{\text{Learn}}$ and $p_1 t \Vdash_s A$, or $p_0 t[s] = \texttt{False}$ in $\mathcal{T}_{\text{Learn}}$ and $p_2 t \Vdash_s B$

(4) $t \Vdash_s A \to B$ if and only if for all u, if $u \Vdash_s A$, then $tu \Vdash_s B$

(5) $t \Vdash_s \forall x A$ if and only if for all numerals n, $tn \Vdash_s A[n/x]$

(6) $t \Vdash_s \exists x A$ if and only for some numeral n, $\pi_0 t[s] = n$ in $\mathcal{T}_{\text{Learn}}$ and $\pi_1 t \Vdash_s A[n/x]$

We define $t \Vdash A$ if and only if $t \Vdash_s A$ for all state constants s.

For the soundness result we shall need this theorem of chapter 3.

THEOREM 4.2.3 . *If A is a closed formula of $\mathcal{L}$ provable in $\mathsf{HA} + \mathsf{EM}_1$, then there exists $t \in \mathcal{T}_{Class}$ such that $t \Vdash A$.*

4.3. Games, Learning and Realizability

In this section, we define the abstract notion of game, its 1-Backtracking version and Tarski games. We also prove our main theorem, connecting learning based realizability and 1-Backtracking Tarski games.

DEFINITION 4.3.1 (GAMES). We define:

(1) A *game* G between two players is a quadruple

$$(V, E_1, E_2, W)$$

where V is a set, E_1, E_2 are subsets of $V \times V$ such that $Dom(E_1) \cap Dom(E_2) = \emptyset$, where $Dom(E_i)$ is the domain of E_i, and W is a set of sequences, possibly infinite, of elements of V. The elements of V are called *positions* of the game; E_1, E_2 are the transition relations respectively for player one and player two: $(v_1, v_2) \in E_i$ means that player i can legally move from the position v_1 to the position v_2.

(2) We define a *play* to be a walk, possibly infinite, in the graph $(V, E_1 \cup E_2)$, i.e. a sequence, possibly void, $v_1 :: v_2 :: \ldots :: v_n :: \ldots$ of elements of V such that $(v_i, v_{i+1}) \in E_1 \cup E_2$ for every i. A play of the form $v_1 :: v_2 :: \ldots :: v_n :: \ldots$ is said to *start from* v_1. A play is said to be *complete* if it is either infinite or is equal to $v_1 :: \ldots :: v_n$ and $v_n \notin Dom(E_1 \cup E_2)$. W is required to be a set of complete plays. If p is a complete play and $p \in W$, we say that player one wins in p. If p is a complete play and $p \notin W$, we say that player two wins in p.

(3) Let P_G be the set of finite plays. Consider a function $f : P_G \to V$. A play $v_1 :: \ldots :: v_n :: \ldots$ is said to be f-correct if $f(v_1 :: \ldots :: v_i) = v_{i+1}$ for every i such that $(v_i, v_{i+1}) \in E_1$. f is said to be a *strategy* for player i if for every play $p = v_1 :: \ldots :: v_n$ such that $v_n \in Dom(E_i)$, $v_1 :: \ldots :: v_n :: f(p)$ is a play.

(4) A *winning strategy* from position v for player one is a strategy $\omega : P_G \to V$ such that every complete ω-correct play $v :: v_1 :: \ldots :: v_n :: \ldots$ belongs to W.

Notation (Concatenation of Sequences). If for $i \in \mathbb{N}, i = 1, \ldots, n$ we have that $p_i = (p_i)_0 :: \ldots :: (p_i)_{n_i}$ is a finite sequence of elements of length n_i, with $p_1 :: \ldots :: p_n$ we denote the sequence

$$(p_1)_0 :: \ldots :: (p_1)_{n_1} :: \ldots :: (p_k)_0 :: \ldots :: (p_k)_{n_k}$$

where $(p_i)_j$ denotes the j-th element of the sequence p_i.

Suppose that $a_1 :: a_2 :: \ldots :: a_n$ is a play of a game G, representing, for some reason, a bad situation for player one (for example, in the game of chess, a_n might be a configuration of the chessboard in which player one has just lost his queen). Then, learnt the lesson, player one might wish to erase some of his moves and come back to the time the play was just, say, a_1, a_2 and choose, say, b_1 in place of a_3; in other words, player one might wish to *backtrack*. Then, the game might go on as $a_1 :: a_2 :: b_1 :: \ldots :: b_m$ and, once again, player one might want to backtrack to, say, $a_1 :: a_2 :: b_1 :: \ldots :: b_i$, with $i < m$, and so on... As there is no learning without remembering, player one must keep in mind the errors made during the play. This is the idea of 1-Backtracking games (for more motivations, we refer the reader to [9]) and here is our definition.

Definition 4.3.2 (1-Backtracking Games). Let $G = (V, E_1, E_2, W)$ be a game.

(1) We define $\mathsf{1back}(G)$ as the game (P_G, E_1', E_2', W'), where:

(2) P_G is the set of finite plays of G

(3)
$$E_2' := \{(p :: a,\ p :: a :: b) \mid p \in P_G, p :: a \in P_G, (a, b) \in E_2\}$$
and
$$E_1' := \{(p :: a,\ p :: a :: b) \mid p \in P_G, p :: a \in P_G, (a, b) \in E_1\} \cup$$
$$\{(p :: a :: q,\ p :: a) \mid p, q \in P_G, p :: a :: q \in P_G, a \in Dom(E_1)$$
$$(q = q' :: d \Rightarrow d \notin Dom(E_2)), p :: a :: q \notin W\};$$

(4) W' is the set of finite complete plays $p_1 :: \ldots :: p_n$ of (P_G, E_1', E_2') such that $p_n \in W$.

Note. The pair $(p :: a :: q,\ p :: a)$ in the definition above of E_2' codifies a *backtracking move* by player one (and we point out that q might be the empty sequence).

Remark. Differently from [9], in which both players are allowed to backtrack, we only consider the case in which only player one is supposed do that (as in [27]). It is not that our results would not hold: we claim that the proofs in this paper would work just as fine for the definition of 1-Backtracking Tarski games given in [9]. However, as noted in [9], any player-one recursive winning strategy in our version of the game can be effectively transformed into a winning strategy for player one in the other version the game. Hence, adding backtracking for the second player does not increase the computational challenge for player one. Moreover, the notion of winner of the game given in [9] is strictly non constructive

and games played by player one with the correct winning strategy may even not terminate. Whereas, with our definition, we can formulate our main theorem as a program termination result: whatever the strategy chosen by player two, the game terminates with the win of player one. This is also the spirit of realizability and hence of this paper: the constructive information must be computed in a finite amount of time, not in the limit.

In the well known Tarski games, there are two players and a formula on the board. The second player - usually called Abelard - tries to show that the formula is false, while the first player - usually called Eloise - tries to show that it is true. Let us see the definition.

DEFINITION 4.3.3 (TARSKI GAMES). Let A be a closed implication and negation free arithmetical formula of $\mathcal{L}$. We define the Tarski game for A as the game $T_A = (V, E_1, E_2, W)$, where:

(1) V is the set of all subformula occurrences of A; that is, V is the smallest set of formulas such that, if either $A \vee B$ or $A \wedge B$ belongs to V, then $A, B \in V$; if either $\forall x A(x)$ or $\exists x A(x)$ belongs to V, then $A(n) \in V$ for all numerals n.

(2) E_1 is the set of pairs $(A_1, A_2) \in V \times V$ such that $A_1 = \exists x A(x)$ and $A_2 = A(n)$, or $A_1 = A \vee B$ and either $A_2 = A$ or $A_2 = B$;

(3) E_2 is the set of pairs $(A_1, A_2) \in V \times V$ such that $A_1 = \forall x A(x)$ and $A_2 = A(n)$, or $A_1 = A \wedge B$ and $A_2 = A$ or $A_2 = B$;

(4) W is the set of finite complete plays $A_1 :: \ldots :: A_n$ such that $A_n = \texttt{True}$.

Note. We stress that Tarski games are defined only for implication-and-negation-free arithmetical formulas. Indeed, $\mathsf{1back}(T_A)$, when A contains implications, would be much more involved and less intuitive (for a definition of Tarski games for every arithmetical formula see for example Lorenzen's [18]).

What we want to show is that if $t \Vdash A$, then t gives to player one a recursive winning strategy in $\mathsf{1back}(T_A)$. The idea of the proof is the following. Suppose we play as player one. Our strategy is relativized to a knowledge state and we start the game by fixing the actual state of knowledge as $\varnothing$. Then we play in the same way as we would do in the Tarski game. For example, if there is $\forall x A(x)$ on the board and $A(n)$ is chosen by player two, we recursively play the strategy given by tn; if there is $\exists x A(x)$ on the board, we calculate $\pi_0 t[\varnothing] = n$ and play $A(n)$ and recursively the strategy given by $\pi_1 t$. If there is $A \vee B$ on the board, we calculate $p_0 t[\varnothing]$, and according as to whether it equals $\texttt{True}$ or $\texttt{False}$, we play the strategy recursively given by $p_1 t$ or $p_2 t$. If there is an atomic formula on the board, if it is true, we win; otherwise we extend the current state with the state $\varnothing \Cup t[\varnothing]$, we backtrack and play with respect to the new state of knowledge and trying to keep as close as possible to the previous game. Eventually, we will reach a state large enough to enable our realizer to give always correct answers and we will win. Let us consider first an example and then the formal definition of the winning strategy for Eloise.

Example (EM_1). Given a predicate P of $\mathcal{T}$, and its boolean negation predicate $\neg P$ (which is representable in $\mathcal{T}$), the realizer E_P of

$$\mathsf{EM}_1 := \forall x.\ \exists y\ P(x,y) \vee \forall y \neg P(x,y)$$

is defined as

$$\lambda\alpha^{\mathtt{N}}\langle \mathsf{X}_P\alpha,\ \langle \Phi_P\alpha,\ \varnothing\rangle,\ \lambda m^{\mathtt{N}}\ \mathsf{Add}_P\alpha m\rangle$$

We now compute a winning strategy for Eloise in the 1-Backtracking game associated to EM_1. According to the rules of the game $\mathsf{1back}(T_{\mathsf{EM}_1})$, Abelard is the first to move and, for some numeral n, chooses the formula

$$\exists y\ P(n,y) \vee \forall y \neg P(n,y)$$

Now is the turn of Eloise and she plays the strategy given by the term

$$\langle \mathsf{X}_P n,\ \langle \Phi_P n,\ \varnothing\rangle,\ \lambda m^{\mathtt{N}}\ \mathsf{Add}_P n m\rangle$$

Hence, she computes $\mathsf{X}_P n[\varnothing] = \chi_P \varnothing n = \mathtt{False}$ (by definition 3.2.7), so she plays the formula

$$\forall y \neg P(n,y)$$

and Abelard chooses m and plays

$$\neg P(n,m)$$

If $\neg P(n,m) = \mathtt{True}$, Eloise wins. Otherwise, she plays the strategy given by

$$(\lambda m^{\mathtt{N}}\ \mathsf{Add}_P n m)m[\varnothing] = \mathsf{add}_P \varnothing n m = \{\langle P,n,m\rangle\}$$

So, the new knowledge state is now $\{\langle P,n,m\rangle\}$ and she backtracks to the formula

$$\exists y\ P(n,y) \vee \forall y \neg P(n,y)$$

Now, by definition 3.2.7, $\mathsf{X}_P n[\{\langle P,n,m\rangle\}] = \mathtt{True}$ and she plays the formula

$$\exists y\ P(n,y)$$

calculates the term

$$\pi_0\langle \Phi_P n,\ \varnothing\rangle[\{\langle P,n,m\rangle\}] = \varphi_P\{\langle P,n,m\rangle\}n = m$$

plays $P(n,m)$ and wins.

Notation. In the following, we shall denote with upper case letters A, B, C closed arithmetical formulas, with lower case letters p, q, r plays of T_A and with upper case letters P, Q, R plays of $\mathsf{1back}(T_A)$ (and all those letters may be indexed by numbers). To avoid confusion with the plays of T_A, plays of $1\mathrm{Back}(T_A)$ will be denoted as $p_1, \ldots, p_n$ rather than $p_1 :: \ldots :: p_n$. Moreover, if $P = q_1, \ldots, q_m$, then $P, p_1, \ldots, p_n$ will denote the sequence $q_1, \ldots, q_m, p_1, \ldots p_n$.

We now define, given a play p of T_A, a term $\rho(p)$, which we call "the realizer associated to p" and which represents the term that should be consulted by Eloise in a position Q, p of the game $\mathsf{1back}(T_A)$.

DEFINITION 4.3.4 . Fix u such that $u \Vdash A$. Let p be a finite play of T_A starting with A. We define by induction on the length of p a term $\rho(p) \in \mathcal{T}_{\text{Class}}$ (read as 'the realizer associated to p') in the following way:

(1) If $p = A$, then $\rho(p) = u$.

(2) If $p = (q :: \exists x B(x) :: B(n))$ and $\rho(q :: \exists x B(x)) = t$, then $\rho(p) = \pi_1 t$.

(3) If $p = (q :: \forall x B(x) :: B(n))$ and $\rho(q :: \forall x B(x)) = t$, then $\rho(p) = tn$.

(4) If $p = (q :: B_0 \wedge B_1 :: B_i)$ and $\rho(q :: B_0 \wedge B_1) = t$, then $\rho(p) = \pi_i t$.

(5) If $p = (q :: B_1 \vee B_2 :: B_i)$ and $\rho(q :: B_1 \vee B_2) = t$, then $\rho(p) = p_i t$.

Given a play $P = Q, q :: B$ of $\mathsf{1back}(T_A)$, we set $\rho(P) = \rho(q :: B)$.

A play P of $\mathsf{1back}(T_A)$ may involve a number of backtracking moves by Eloise. In the winning strategy we are going to define, each time Eloise backtracks, she must extend the current state of knowledge by means of a realizer. In the above definition, we explain how Eloise calculates the state associated to P.

DEFINITION 4.3.5 . Fix u such that $u \Vdash A$. Let ρ be as in definition 4.3.4 and P be a finite play of $\mathsf{1back}(T_A)$ starting with A. We define by induction on the length of P a state $\Sigma(P)$ (read as 'the state associated to P') in the following way:

(1) If $P = A$, then $\Sigma(P) = \varnothing$.

(2) If $P = (Q, p :: B, p :: B :: C)$ and $\Sigma(Q, p :: B) = s$, then $\Sigma(P) = s$.

(3) If $P = (Q, p :: B :: q, p :: B)$ and $\Sigma(Q, p :: B :: q) = s$ and $\rho(Q, p :: B :: q) = t$, then if $t : \mathtt{S}$, then $\Sigma(P) = s \Cup t[s]$, else $\Sigma(P) = s$.

We are now in a position to define a winning strategy for Eloise. Given a play Q, p of $\mathsf{1back}(T_A)$, she computes the state associated to Q, p and then calls the realizer associated to p, which returns to her the next move to be performed.

DEFINITION 4.3.6 (WINNING STRATEGY FOR $\mathsf{1back}(T_A)$). Fix u such that $u \Vdash A$. Let ρ and Σ be respectively as in definitions 4.3.4 and 4.3.5. We define a function ω from the set of finite plays of $\mathsf{1back}(T_A)$ to set of finite plays of T_A; ω is intended to be a recursive winning strategy from A for player one in $\mathsf{1back}(T_A)$.

(1) If $\rho(P, q :: \exists x B(x)) = t$, $\Sigma(P, q :: \exists x B(x)) = s$ and $(\pi_0 t)[s] = n$, then

$$\omega(P, q :: \exists x B(x)) = q :: \exists x B(x) :: B(n)$$

(2) If $\rho(P, q :: B \vee C) = t$ and $\Sigma(P, q :: B \vee C) = s$, then if $(p_0 t)[s] = \mathtt{True}$ then

$$\omega(P, q :: B \vee C) = q :: B \vee C :: B$$

else

$$\omega(P, q :: B \vee C) = q :: B \vee C :: C$$

(3) If A_n is atomic, $A_n = \texttt{False}$, $\rho(P, A_1 :: \cdots :: A_n) = t$ and $\Sigma(P, A_1 :: \cdots :: A_n) = s$, then
$$\omega(P, A_1 :: \cdots :: A_n) = A_1 :: \cdots :: A_i$$
where i is equal to the smallest $j < n$ such that $\rho(A_1 :: \cdots :: A_j) = w$ and either
$$A_j = \exists x C(x) \wedge A_{j+1} = C(n) \wedge (\pi_0 w)[s \Cup t[s]] \neq n$$
or
$$A_j = B_1 \vee B_2 \wedge A_{j+1} = B_1 \wedge (p_0 w)[s \Cup t[s]] = \texttt{False}$$
or
$$A_j = B_1 \vee B_2 \wedge A_{j+1} = B_2 \wedge (p_0 w)[s \Cup t[s]] = \texttt{True}$$
If such j does not exist, we set $i = n$.

(4) In the other cases, $\omega(P, q) = q$.

LEMMA 4.3.1 . *Suppose $u \Vdash A$ and ρ, Σ, ω as in definition 4.3.6. Let Q be a finite ω-correct play of $\mathsf{1back}(T_A)$ starting with A, $\rho(Q) = t$, $\Sigma(Q) = s$. If $Q = Q', q' :: B$, then $t \Vdash_s B$.*

PROOF. By a straightforward induction on the length of Q.

(1) If $Q = A$, then $t = \rho(Q) = u \Vdash_s A$.

(2) If $Q = P, q :: \exists x B(x), q :: \exists x B(x) :: B(n)$, then let $t' = \rho(P, q :: \exists x B(x))$. By definition of Σ, $s = \Sigma(P, q :: \exists x B(x))$. Since Q is ω-correct and $(q :: \exists x B(x), q :: \exists x B(x) :: B(n)) \in E_1$, we have $\omega(P, q :: \exists x B(x)) = q :: \exists x B(x) :: B(n)$ and so $n = (\pi_0 t')[s]$. Moreover, by definition of ρ, $t = \pi_1 t'$; by induction hypothesis, $t' \Vdash_s \exists x B(x)$; so, $t = \pi_1 t' \Vdash_s B(n)$.

(3) If $Q = P, q :: B \vee C, q :: B \vee C :: B$, then let $t' = \rho(P, q :: B \vee C)$. By definition of Σ, $s = \Sigma(P, q :: B \vee C)$. Since Q is ω-correct and $(q :: B \vee C, q :: B \vee C :: B) \in E_1$, we have $\omega(P, q :: B \vee C) = q :: B \vee C :: B$ and so $(p_0 t')[s] = \texttt{True}$. Moreover, by definition of ρ, $t = p_1 t'$; by induction hypothesis, $t' \Vdash_s B \vee C$; so, $t = p_1 t' \Vdash_s B$. The other case is analogous.

(4) If $Q = P, q :: \forall x B(x), q :: \forall x B(x) :: B(n)$, then let $t' = \rho(P, q :: \forall x B(x))$. By definition of Σ, $s = \Sigma(P, q :: \forall x B(x))$. By definition of ρ, $t = t'n$; by induction hypothesis, $t' \Vdash_s \forall x B(x)$; hence, $t = t'n \Vdash_s B(n)$.

(5) If $Q = P, q :: B \wedge C, q :: B \wedge C :: B$, then let $t' = \rho(P, q :: B \wedge C)$. By definition of Σ, $s = \Sigma(P, q :: B \wedge C)$. By definition of ρ, $t = \pi_0 t'$; by induction hypothesis, $t' \Vdash_s B \wedge C$; hence, $t = \pi_0 t' \Vdash_s B$. The other case is analogous.

(6) If $Q = P, A_1 :: \cdots :: A_n, A_1 :: \cdots :: A_i$, $i < n$, A_n atomic, then $A_1 = A$. Furthermore, if $\Sigma(P, A_1 :: \cdots :: A_n) = s'$ and $t' = \rho(P, A_1 :: \cdots :: A_n)$, then $s = s' \Cup t'[s']$. Let $t_j = \rho(A_1 :: \cdots :: A_j)$, for $j = 1, \ldots, i$. We prove by induction on j that $t_j \Vdash_s A_j$, and hence the thesis. If $j = 1$, then $t_1 = \rho(A_1) = \rho(A) = u \Vdash_s A = A_1$. If $j > 1$, by induction hypothesis $t_k \Vdash_s A_k$, for every $k < j$. If either $A_{j-1} = \forall x C(x)$ or $A_{j-1} = C_0 \wedge C_1$, then either $t_j = t_{j-1} n$ and $A_j = C(n)$, or $t_j = \pi_m t_{j-1}$

and $A_j = C_m$: in both cases, we have $t_j \Vdash_s A_j$, since $t_{j-1} \Vdash_s A_{j-1}$. Therefore, by definition of ω and i and the ω-*correctness* of Q, the remaining possibilities are that either $A_{j-1} = \exists x C(x)$, $A_j = C(n)$, $t_j = \pi_1 t_{j-1}$, with $(\pi_0 t_{j-1})[s] = n$; or $A_{j-1} = C_1 \vee C_2$, $A_j = C_m$, $t_j = p_m t_{j-1}$ and $(p_0 t_{j-1})[s] = \texttt{True}$ if and only if $m = 1$; in both cases, we have $t_j \Vdash_s A_j$.

□

THEOREM 4.3.1 (SOUNDNESS THEOREM). *Let A be a closed negation and implication free arithmetical formula. Suppose that $u \Vdash A$ and consider the game $\mathsf{1back}(T_A)$. Let ω be as in definition 4.3.6. Then ω is a recursive winning strategy from A for player one.*

PROOF. We begin by showing that there is no infinite ω-correct play.
Let $P = p_1, \dots, p_n, \dots$ be, for the sake of contradiction, an infinite ω-correct play, with $p_1 = A$. Let $A_1 :: \cdots :: A_k$ be the *longest* play of T_A such that there exists j such that for every $n \geq j$, p_n is of the form $A_1 :: \cdots :: A_k :: q_n$. $A_1 :: \cdots :: A_k$ is well defined, because: p_n is of the form $A :: q'_n$ for every n; the length of p_n is at most the degree of the formula A; the sequence of maximum length is unique because any two such sequences are one the prefix of the other, and therefore are equal. Moreover, let $\{n_i\}_{i\in\mathbb{N}}$ be the infinite increasing sequence of all indexes n_i such that p_{n_i} is of the form $A_1 :: \cdots :: A_k :: q_{n_i}$ and $p_{n_i+1} = A_1 :: \cdots :: A_k$ (indeed, $\{n_i\}_{i\in\mathbb{N}}$ must be infinite: if it were not so, then there would be an index j' such that for every $n \geq j'$, $p_n = A_1 :: \cdots :: A_k :: A_{k+1} :: q$, violating the assumption on the maximal length of $A_1 :: \cdots :: A_k$). A_k, if not atomic, is a disjunction or an existential statement.
Let now $s_i = \Sigma(p_1, \dots, p_i)$ and $t = \rho(A_1 :: \cdots :: A_k)$. For every i, $s_i \leq s_{i+1}$, by definition of Σ. There are three cases:

1) $A_k = \exists x B(x)$. Then, by the Stability Theorem (Theorem 4.2.1), there exists m such that for every a, if $n_a \geq m$, then $(\pi_0 t)[s_{n_a}] = (\pi_0 t)[s_m]$. Let

$$h := (\pi_0 t)[s_{n_a} \Cup t_1[s_{n_a}]] = (\pi_0 t)[s_{n_a+1}]$$

where $t_1 = \rho(p_1, \dots, p_{n_a})$. So let a be such that $n_a \geq m$; then

$$p_{n_a+2} = \omega(p_1, \dots, p_{n_a+1}) = \omega(p_1, \dots, A_1 :: \cdots :: A_k) = A_1 :: \cdots :: A_k :: B(h)$$

Moreover, by hypothesis, and since $p_{n_a+1} = A_1 :: \cdots :: A_k$, we have

$$p_{n_{(a+1)}} = A_1 :: \cdots :: A_k :: q_{n_{(a+1)}} = A_1 :: \cdots :: A_k :: B(h) :: q'$$

for some q' and $p_{n_{(a+1)}+1} = A_1 :: \cdots :: A_k$: contradiction, since

$$h = (\pi_0 t)[s_{n_a+1}] = (\pi_0 t)[s_{n_{(a+1)}+1}] = (\pi_0 t)[s_{n_{(a+1)}} \Cup t_2[s_{n_{(a+1)}}]]$$

where $t_2 = \rho(p_1, \dots, p_{n_{(a+1)}})$, whilst $h \neq (\pi_0 t)[s_{n_{(a+1)}} \Cup t_2[s_{n_{(a+1)}}]]$ should hold, by definition of ω (point (3)).

2) $A_k = B \vee C$. This case is totally analogous to the preceding.

3) A_k is atomic. Then, for every $n \geq j$, $p_n = A_1 :: \cdots :: A_k$. So, for every $n \geq j$, $s_{n+1} = s_n \Cup t[s_n]$ and hence, by Theorem 4.2.2 there exists $m \geq j$ such that $t[s_m] = \varnothing$. But $t \Vdash_{s_m} A_k$, by Lemma 4.3.1; hence, A_k must equal $\texttt{True}$, and so it is impossible that

$(p_m, p_{m+1}) = (A_1 :: \cdots :: A_k, A_1 :: \cdots :: A_k) \in E_1'$: contradiction.
Let now $p = p_1, \ldots, p_n$ be a complete finite ω-correct play. p_n must equal $B_1 :: \cdots :: B_k$, with B_k atomic and $B_k = \texttt{True}$: otherwise, p wouldn't be complete, since player one would lose the play p_n in T_A and hence would be allowed to backtrack by definition 4.3.2.

□

4.4. Examples

In this section we include two natural deduction classical proofs of two simple combinatorial statements, using only Excluded Middle for semi-decidable statements, then we extract a constructive content using our realizability semantics. For each of them, we interpret the program we shall extract using our game interpretation of the learning based realizability semantics.

4.4.1. Minimum Principle for Functions over Natural Numbers. The minimum principle states that every function f over natural numbers has a minimum value, i.e. there exists a $f(n) \in \mathbb{N}$ such that for every $m \in \mathbb{N}$ $f(m) \geq f(n)$. We can prove this principle in $\mathsf{HA} + \mathsf{EM}_1$, for any f in the language. We assume $P(y, x) \equiv f(x) < y$, but, in order to enhance readability, we will write $f(x) < y$ rather than the obscure $P(y, x)$. We define:
$Lessef(n) := \exists \alpha f(\alpha) \leq n$
$Lessf(n) := \exists \alpha f(\alpha) < n$
$Notlessf(n) := \forall \alpha f(\alpha) \geq n$
Then we formulate - in equivalent form - the minimum principle as:

$$Hasminf := \exists y.\ Notlessf(y) \wedge Lessef(y)$$

The informal argument goes as follows. We prove by induction on n that for every k, if $f(k) \leq n$, then f has minimum value. If $n = 0$, we just observe that $f(k) \leq 0$, implies $f(k)$ is the minimum value of f. Suppose now $n > 0$. If $Notlessf(f(k))$ holds true, we are done, $f(k)$ is the minimum of f. Otherwise, $Lessf(f(k))$ holds, and hence $f(\alpha) < f(k) \leq n$ for some α given by an oracle. Hence $f(\alpha) \leq f(k) - 1 \leq n - 1$ and we conclude that f has a minimum value by induction hypothesis.

Now we give the formal proofs, which are natural deduction trees, decorated with terms of $\mathcal{T}_{\text{Class}}$, as formalized in chapter 3. We first prove that $\forall n.\ (Lessef(n) \to Hasminf) \to (Lessef(\mathsf{S}(n)) \to Hasminf)$ holds.

$$\dfrac{\dfrac{\dfrac{\dfrac{E_P : \forall n.\ Notlessf(\mathsf{S}(n)) \vee Lessf(\mathsf{S}(n))}{E_P n : Notlessf(\mathsf{S}(n)) \vee Lessf(\mathsf{S}(n))} \quad \begin{matrix}[Notlessf(\mathsf{S}(n))] \\ T_1 \\ Hasminf\end{matrix} \quad \begin{matrix}[Lessf(\mathsf{S}(n))] \\ T_2 \\ Hasminf\end{matrix}}{D : Hasminf}}{\lambda w_2 D : Lessef(\mathsf{S}(n)) \to Hasminf}}{\lambda w_1 \lambda w_2 D : (Lessef(n) \to Hasminf) \to (Lessef(\mathsf{S}(n)) \to Hasminf)}}{\lambda n \lambda w_1 \lambda w_2 D : \forall n (Lessef(n) \to Hasminf) \to (Lessef(\mathsf{S}(n) \to Hasminf)}$$

where for lack of space the term D is defined later, T_1 is the tree

$$\dfrac{\dfrac{v_1 : Notlessf(\mathsf{S}(n)) \qquad w_2 : Lessef(\mathsf{S}(n))}{\langle v_1, w_2 \rangle : Notlessf(\mathsf{S}(n)) \wedge Lessef(\mathsf{S}(n))}}{\langle \mathsf{S}(n), \langle v_1, w_2 \rangle \rangle : Hasminf}$$

and T_2 is the tree

$$\dfrac{v_2 : [Lessf(\mathsf{S}(n))] \qquad \dfrac{w_1 : [Lessef(n) \to Hasminf] \qquad \dfrac{\dfrac{[x_2 : f(z) < \mathsf{S}(n)]}{x_2 : f(z) \leq n}}{\langle z, x_2\rangle : Lessef(n)}}{w_1\langle z, x_2\rangle : Hasminf}}{w_1\langle \pi_0 v_2, \pi_1 v_2\rangle : Hasminf}$$

We prove now that $Lessef(0) \to Hasminf$

$$\dfrac{w : [Lessef(0)] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x_1 : [f(z) \leq 0]}{x_1 : f(z) = 0}}{x_1 : f(\alpha) \geq f(z)}}{\lambda\alpha x_1 : Notlessf(f(z))} \qquad \dfrac{\varnothing : f(z) \leq f(z)}{\langle z, \varnothing\rangle : Lessef(f(z))}}{\langle \lambda\alpha x_1, \langle z, \varnothing\rangle\rangle : Notlessf(f(z)) \wedge Lessef(f(z))}}{\langle f(z), \langle \lambda\alpha x_1, \langle z, \varnothing\rangle\rangle\rangle : Hasminf}}{\langle f(\pi_0 w), \langle \lambda\alpha\pi_1 w, \langle \pi_0 w, \varnothing\rangle\rangle\rangle : Hasminf}}{F := \lambda w\langle f(\pi_0 w), \langle \lambda\alpha\pi_1 w, \langle \pi_0 w, \varnothing\rangle\rangle\rangle : Lessef(0) \to Hasminf}$$

Therefore we can conclude with the induction rule that

$$\lambda\alpha^{\mathtt{N}}\ \mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)\alpha : \forall x.Lessef(x) \to Hasminf$$

And now the thesis:

$$\dfrac{\dfrac{\varnothing : f(0) \leq f(0)}{\langle 0, \varnothing\rangle : Lessef(f(0))} \qquad \dfrac{\lambda\alpha^{\mathtt{N}}\ \mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)\alpha : \forall x.Lessef(x) \to Hasminf}{\mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0) : Lessef(f(0)) \to Hasminf}}{M := \mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0)\langle 0, \varnothing\rangle : Hasminf}$$

Let us now define

$$D := \mathtt{if}\ \mathsf{X}_P\mathsf{S}(n)\ \mathtt{then}\ w_1\langle \Phi_P\mathsf{S}(n), \varnothing\rangle\ \mathtt{else}\ \langle \mathsf{S}(n), \langle \lambda\beta\ (\mathsf{Add}_P)\mathsf{S}(n)\beta, w_2\rangle\rangle$$

Let s be a state and let us consider M, the realizer of $Hasminf$, in the base case of the recursion and after in its general form during the computation: $\mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0)\langle m, \varnothing\rangle[s]$. If $f(0) = 0$,

$$M[s] = \mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0)\langle 0, \varnothing\rangle[s] =$$
$$= F\langle 0, \varnothing\rangle = \langle f(0), \langle \lambda\alpha\varnothing, \langle 0, \varnothing\rangle\rangle$$

If $f(0) = \mathsf{S}(n)$, we have two other cases. If $\chi_P s\mathsf{S}(n) = \mathtt{True}$, then

$$\mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)\mathsf{S}(n)\langle m, \varnothing\rangle[s] =$$
$$= (\lambda n\lambda w_1\lambda w_2 D)n(\mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)n)\langle m, \varnothing\rangle[s] =$$
$$= \mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)n\langle \Phi_P(\mathsf{S}(n)), \varnothing\rangle[s]$$

If $\chi_P s\mathsf{S}(n) = \mathtt{False}$, then

$$\mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)\mathsf{S}(n)\langle m, \varnothing\rangle[s] =$$
$$= (\lambda n\lambda w_1\lambda w_2 D)n(\mathsf{R}F(\lambda n\lambda w_1\lambda w_2 D)n)\langle m, \varnothing\rangle[s] =$$
$$= \langle \mathsf{S}(n), \langle \lambda\beta\ (\mathtt{add}_P)s\mathsf{S}(n)\beta, \langle m, \varnothing\rangle\rangle\rangle$$

In the first case, the minimum value of f has been found. In the second case, the operator R, starting from $\mathsf{S}(n)$, recursively calls itself on n; in the third case, it reduces to its normal form. From these equations, we easily deduce the behavior of the realizer of $Hasminf$. In a pseudo imperative programming language, for the witness of $Hasminf$ we would write:

$n := f(0);$

`while` $(\chi_P sn = \texttt{True}, i.e.\ \exists m\ such\ that\ f(m) < n \in s)$
`do` $n := n - 1$;
`return` n;

Hence, when $f(0) > 0$, we have, for some numeral k

$$M[s] = \langle k, \langle \lambda\beta\ (\mathsf{add}_P)sk\beta, \langle \varphi_P sk, \varnothing\rangle\rangle\rangle$$

It is clear that k is the minimum value of f, according to the partial information provided by s about f, and that $f(\varphi_P sk) \leq k$. If s is sufficiently complete, then k is the true minimum of f.
The normal form of the realizer M of $Hasminf$ is so simple that we can immediately extract the winning strategy ω for the 1-Backtraking version of the Tarski game for $Hasminf$. Suppose the current state of the game is s. If $f(0) = 0$, Eloise chooses the formula

$$Notlessf(0) \wedge Lessef(0)$$

and wins. If $f(0) > 0$, she chooses, for k defined as above,

$$Notlessf(k) \wedge Lessef(k) = \forall\alpha\ f(\alpha) \geq k \wedge \exists\alpha\ f(\alpha) \leq k$$

If Abelard chooses $\exists\alpha\ f(\alpha) \leq k$, she wins, because she responds with $f(\varphi_P sk) \leq k$, which holds. Suppose hence Abelard chooses

$$\forall\alpha\ f(\alpha) \geq k$$

and then $f(\beta) \geq k$. If it holds, Eloise wins. Otherwise, she adds to the current state s

$$(\lambda\beta\ (\mathsf{add}_P)sk\beta)\beta = (\mathsf{add}_P)sk\beta = \{f(\beta) < k\}$$

and backtracks to $Hasminf$ and then plays again. This time, she chooses

$$Notlessf(f(\beta)) \wedge Lessef(f(\beta))$$

(using $f(\beta)$, which was Abelard's counterexample to the minimality of k and is smaller than her previous choice for the minimum value). After at most $f(0)$ backtrackings, she wins.

4.4.2. Coquand's Example. We investigate now an example - due to Coquand - in our framework of realizability. We want to prove that for every function over natural numbers and for every $a \in \mathbb{N}$ there exists $x \in \mathbb{N}$ such that $f(x) \leq f(x + a)$. Thanks to the minimum principle, we can give a very easy classical proof:

$$\dfrac{Hasminf \qquad \dfrac{\dfrac{[Notlessf(\mu) \wedge Lessef(\mu)]}{Lessef(\mu)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{[Notlessf(\mu) \wedge Lessef(\mu)]}{Notlessf(\mu)}}{f(z+a) \geq \mu} \qquad [f(z) \leq \mu]}{f(z) \leq f(z+a)}}{\exists x f(x) \leq f(x+a)}}{\forall a \exists x f(x) \leq f(x+a)}}{\forall a \exists x f(x) \leq f(x+a)}$$

The extracted realizer is

$$\lambda a \langle \pi_0\pi_1\pi_1 M, \pi_0\pi_1 M(\pi_0\pi_1\pi_1 M + a) \uplus \pi_1\pi_1\pi_1 h\rangle$$

where M is the realizer of $Hasminf$. $m := \pi_0\pi_1\pi_1 M[s]$ is a point the purported minimum value $\mu := \pi_0 M$ of f is attained at, accordingly to the information in the state s (i.e. $f(m) \leq \mu$). So, if Abelard chooses

$$\exists x\ f(x) \leq f(x+a)$$

Eloise chooses

$$f(m) \leq f(m+a)$$

We have to consider the term

$$U[s] := \pi_0\pi_1 M(\pi_0\pi_1\pi_1 M + a) \Cup \pi_1\pi_1\pi_1 M[s]$$

which updates the current state s. Surely, $\pi_1\pi_1\pi_1 M[s] = \varnothing$. $\pi_0\pi_1 M[s]$ is equal either to $\lambda\beta\ (\mathsf{add}_P)s\mu\beta$ or to $\lambda\alpha\varnothing$. So, what does $U[s]$ actually do? We have:

$$U[s] = \pi_0\pi_1 M(\pi_0\pi_1\pi_1 M + a)[s] = \pi_0\pi_1 M(m+a)[s]$$

with either $\pi_0\pi_1 M(m+a)[s] = \varnothing$ or

$$\pi_0\pi_1 M(m+a)[s] = \{f(m+a) < f(m)\}$$

So $U[s]$ tests if $f(m+a) < f(m)$; if it is not the case, Eloise wins, otherwise she enlarges the state s, including the information $f(m+a) < f(m)$ and backtracks to $\exists x f(x) \leq f(x+a)$. Starting from the state $\varnothing$, after $k+1$ backtrackings, it will be reached a state s', which will be of the form $\{f((k+1)a) < f(ka), \ldots, f(2a) < f(a), f(a) < f(0)\}$ and Eloise will play $f((k+1)a) \leq f((k+1)a + a)$. Hence, the extracted algorithm for Eloise's witness is the following:

$n := 0$;
`while` $f(n) > f(n+a)$
`do` $n := n + a$;
`return` n;

4.5. Total Recursive Learning-Based Realizability

The realizability notion introduced in definition 4.2.4 is very interesting from the constructive point of view. But precisely for that reason, the system $\mathcal{T}_{\text{Class}}$ fails to realize every formula for which an Eloise recursive winning strategy exists in its associated 1-Backtracking Tarski game, as the following theorem implies:

THEOREM 4.5.1 (INCOMPLETENESS OF $\mathcal{T}_{\text{CLASS}}$). *There is a Π^0_2 arithmetical sentence A such that Eloise has recursive winning strategy in $\mathsf{1back}(T_A)$, but no term of system $\mathcal{T}_{Class}$ realizes A.*

PROOF. Take any total recursive function $f : \mathtt{N} \to \mathtt{N}$ not representable by any term of system T of type $\mathtt{N} \to \mathtt{N}$. Let n be the code of f in the enumeration of Turing machines assumed by Kleene's primitive recursive predicate $Txyz$. Then the formula $A := \forall y \exists z Tnyz$ asserts the totality of f and hence it is true. Clearly, Eloise has a winning recursive strategy in $\mathsf{1back}(T_A)$: for any y, she may backtrack until she finds an m such that $Tnym$ holds. Suppose, along the way of contradiction, that for some t of system $\mathcal{T}_{\text{Class}}$, $t \Vdash A$. Then, as proven in chapter 5, there exists a term $u : \mathtt{N} \to \mathtt{N}$ of system T such that, for every numeral

l, $Tnl(ul)$ holds. From this, it easily follows that f can be coded by a term of system T, which is a contradiction.

□

The only way around this issue, which is the purpose of this section, is to extend our notion of realizability and increase the computational power of our realizers, in order to be able to represent any partial recursive function and in particular every recursive strategy of 1-Backtracking Tarski games. So, we choose to add to our calculus a fixed point combinator Y, such that for every term $u : A \to A$, $\mathsf{Y}u = u(\mathsf{Y}u)$, getting the full power of $\mathcal{PCF}$ (see for example Gunter [25]).

Definition 4.5.1 (Systems $\mathcal{PCF}_{\text{Class}}$ and $\mathcal{PCF}_{\text{Learn}}$). We define $\mathcal{PCF}_{\text{Class}}$ and $\mathcal{PCF}_{\text{Learn}}$ to be, respectively, the extensions of $\mathcal{T}_{\text{Class}}$ and $\mathcal{T}_{\text{Learn}}$ obtained by adding for every type A a constant Y_A of type $(A \to A) \to A$ and a new equality axiom $\mathsf{Y_A}u = u(\mathsf{Y_A}u)$ for every term $u : A \to A$.

Since in $\mathcal{PCF}_{\text{Class}}$ there is a schema for unbounded iteration, properties like convergence do not hold anymore, for terms may even not have a normal form. So we have to *ask* our realizers to be convergent. Hence, for each type A of $\mathcal{PCF}_{\text{Class}}$ we define a set $\|A\|$ of terms $u : A$ which we call the set of *stable terms* of type A. We define stable terms by lifting the notion of convergence from atomic types to arrow and product types.

Definition 4.5.2 (Convergence for $\mathcal{PCF}_{\text{Class}}$). Assume that $\{s_i\}_{i\in\mathbb{N}}$ is a w.i. sequence of state constants, and $u, v \in \mathcal{PCF}_{\text{Class}}$.

(1) u converges in $\{s_i\}_{i\in\mathbb{N}}$ if there exists a normal form v such that $\exists i \forall j \geq i . u[s_j] = v$ in $\mathcal{PCF}_{\text{Learn}}$.
(2) u converges if u converges in every w.i. sequence of state constants.

Definition 4.5.3 (Stable Terms). Let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of states and $s \in \mathbb{S}$. Assume A is a type. We define a set $\|A\|$ of terms $t \in \mathcal{PCF}_{\text{Class}}$ of type A, by induction on A.

(1) $\|\mathtt{S}\| = \{t : \mathtt{S} \mid t \text{ converges}\}$

(2) $\|\mathtt{N}\| = \{t : \mathtt{N} \mid t \text{ converges}\}$

(3) $\|\mathtt{Bool}\| = \{t : \mathtt{Bool} \mid t \text{ converges}\}$

(4) $\|A \times B\| = \{t : A \times B \mid \pi_0 t \in \|A\|, \pi_1 t \in \|B\|\}$

(5) $\|A \to B\| = \{t : A \to B \mid \forall u \in \|A\|, tu \in \|B\|\}$

If $t \in \|A\|$, we say that t is a *stable* term of type A.

Now we extend the notion of realizability with respect to $\mathcal{PCF}_{\text{Class}}$ and $\mathcal{PCF}_{\text{Learn}}$.

Definition 4.5.4 (Total Recursive Learning-Based Realizability). Assume s is a state constant, $t \in \mathcal{PCF}_{\text{Class}}$ is a closed term of state $\varnothing$, $A \in \mathcal{L}$ is a closed formula, and $t \in \|[A]\|$. Let $\vec{t} = t_1, \ldots, t_n : \mathtt{N}$.

(1) $t \Vdash_s P(\vec{t})$ if and only if $t[s] = \varnothing$ in $\mathcal{PCF}_{\text{Learn}}$ implies $P(\vec{t}) = \mathtt{True}$

(2) $t \Vdash_s A \wedge B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$

(3) $t \Vdash_s A \vee B$ if and only if either $p_0 t[s] = \mathtt{True}$ in $\mathcal{PCF}_{\text{Learn}}$ and $p_1 t \Vdash_s A$, or $p_0 t[s] = \mathtt{False}$ in $\mathcal{PCF}_{\text{Learn}}$ and $p_2 t \Vdash_s B$

(4) $t \Vdash_s A \to B$ if and only if for all u, if $u \Vdash_s A$, then $tu \Vdash_s B$

(5) $t \Vdash_s \forall x A$ if and only if for all numerals n, $tn \Vdash_s A[n/x]$

(6) $t \Vdash_s \exists x A$ if and only for some numeral n, $\pi_0 t[s] = n$ in $\mathcal{PCF}_{\text{Learn}}$ and $\pi_1 t \Vdash_s A[n/x]$

We define $t \Vdash A$ if and only if $t \Vdash_s A$ for all state constants s.

We observe that theorem 4.2.2 holds as well for the stable terms of $\mathcal{PCF}_{\text{Class}}$, for it is a consequence of the Stability theorem 4.2.1. Hence, the Soundness theorem 4.3.1, which depends only on the definition of realizability, stability and theorem 4.2.2, also holds for realizers of $\mathcal{PCF}_{\text{Class}}$. That is, we have

Theorem 4.5.2 (Soundness Theorem ($\mathcal{PCF}_{\text{Class}}$)). *Let A be a closed negation-and-implication free arithmetical formula. Suppose that $u \in \mathcal{PCF}_{Class}$ and $u \Vdash A$ and consider the game $\mathsf{1back}(T_A)$. Let ω be as in definition 4.3.6. Then ω is a recursive winning strategy from A for player one.*

4.6. Completeness

4.6.1. Idea of the Proof. In this section, we prove our completeness theorem: if an implication-and-negation-free arithmetical formula has a winning recursive strategy in its associated 1-Backtracking Tarski game, then it is realizable by a term of $\mathcal{PCF}_{\text{Class}}$.

The idea of the proof follows naturally from the very meaning of learning based realizability. In order to realize a formula, one has to provide in the first place a Kleene-Kreisel-style realizer of the formula, *recursive in an oracle* for the Halting problem. This corresponds to the fact that the terms of $\mathcal{T}_{\text{Class}}$ contain symbols for non computable functions which are in the same Turing degree of the aforementioned oracle. That is why one can see learning based realizability as a way of "programming with non computable functions". Hence, one would like to apply directly Berardi et al. [9] result: given an implication-and-negation-free arithmetical formula, if there exists a recursive winning strategy for Eloise in its associated 1-Backtracking Tarski game, then there also exists a winning strategy for Eloise in its associated Tarski game, *recursive in an oracle* for the Halting problem.

However, that result is not enough for our purposes. According to learning based realizability, together with an oracle-equipped Kleene-Kreisel-style realizer, one has also to provide an effective method for learning oracle values in a convergent way and show that

the realizer is always defined, whatever oracle approximations are used. Hence, we have to refine Berardi et al. result and prove that oracle values can be learned by counterexamples and the program is not "perturbed" by the oracle approximations used. More precisely, we show that the ineffective oracle strategy given in [9] can be made more effective by using the novel ideas of learning based realizability: we first approximate the strategy by allowing it to use in the computations only approximated oracles; then we show that good enough approximations can be attained by a process of intelligent learning by counterexamples. One this task will be accomplished, the completeness theorem will follow just by formalizing the argument.

We now give an informal overview of the construction to be carried out. This should serve the reader as a guide to the next technical sections. Suppose that ω is a recursive winning strategy for Eloise in $\mathsf{1back}(T_A)$. We start by describing a winning strategy for Eloise in T_A, which is recursive in some oracle for the Halting problem. We begin with some terminology.

Definition 4.6.1 (Improvable, Optimal Plays). We say that a ω-correct play

$$Q, A_0 :: \ldots :: A_i$$

of $\mathsf{1back}(T_A)$, with A_i of the form $\exists x B$ or $B \vee C$, is *improvable* if there exists a ω-correct play of $\mathsf{1back}(T_A)$ of the form

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

and we call this latter play an *improvement* of the former. Moreover, a play is said to be *optimal* if it is not improvable.

The reason why a play

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

is called an improvement of $Q, A_0 :: \ldots :: A_i$ is that the former gives more information to the strategy ω in order to choose the next move for Eloise. Moreover, if $Q, A_0 :: \ldots :: A_i$ is optimal, whatever ω-correct continuation of the game we may consider, ω will not backtrack to $A_0 :: \ldots :: A_i$ anymore. Since any such continuation will extend the play

$$Q, A_0 :: \ldots :: A_i, A_0 :: \ldots :: A_i :: A_{i+1}$$

where

$$\omega(Q, A_0 :: \ldots :: A_i) = A_0 :: \ldots :: A_i :: A_{i+1}$$

the choice of A_{i+1} operated by ω is the best possible.

The oracle X_E that we consider answers to questions of the form: is the play $Q, A_0 :: \ldots :: A_i$ improvable? To facilitate computations we also consider an oracle Φ_E which given the code of a play $Q, A_0 :: \ldots :: A_i$ returns the code of a play

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

whenever $Q, A_0 :: \ldots :: A_i$ is improvable and returns a dummy code otherwise. Observe that the two oracles X_E and Φ_E are of the same Turing degree, so exactly one of them would suffice. Furthermore, observe that one can define a program taking as input a code

of an improvable play $Q, A_0 :: \ldots :: A_i$ and returning the code of an optimal improvement of the form

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

(just iterate Φ_E).

Suppose now that $A_0 :: \ldots :: \exists x B$ is a position in the game T_A. How should Eloise move? The idea, coming from [9], is to compute, using our oracles, an optimal play of $\mathsf{1back}(T_A)$ of the form

$$Q, A_0 :: \ldots :: \exists x B$$

Then, Eloise should respond by first computing

$$\omega(Q, A_0 :: \ldots :: \exists x B) = A_0 :: \ldots :: \exists x B :: B(n)$$

and then choosing the formula $B(n)$. The idea is that $B(n)$ is a good choice, since no backtracking to $A_0 :: \ldots :: \exists x A$ is ever to be done, following the strategy ω.

More precisely, Eloise, while playing, simultaneously constructs a sequence of plays $Q_0, \ldots, Q_k$ of $\mathsf{1back}(T_A)$ in the following way. She first defines Q_0 to be an optimal improvement of A. Then, suppose $k > 0$ and that she is in the position

$$A_0 :: \ldots :: A_k$$

of the game T_A and has constructed a sequence of plays $Q_0, \ldots, Q_k$ of $\mathsf{1back}(T_A)$ such that

i) each play Q_i is of the form $Q'_i, A_0 :: \ldots :: A_i$;

ii) For all $i < k$, Q_{i+1} extends Q_i;

iii) For all i, Q_i is optimal;

iv) For all i, Q_i is ω-correct.

Then if Abelard has to move and chooses A_{k+1}, Eloise defines Q_{k+1} as an optimal improvement of

$$Q_k, A_0 :: \ldots :: A_k :: A_{k+1}$$

which she can compute using the oracles. If Eloise has to move, she computes

$$\omega(Q_k) = A_0 :: \ldots :: A_k :: A_{k+1}$$

she chooses as next move A_{k+1} and she defines Q_{k+1} as an optimal improvement of

$$Q_k, A_0 :: \ldots :: A_k :: A_{k+1}$$

which she can again compute using the oracles. It is clear that the sequence $Q_1, \ldots, Q_{k+1}$ still satisfies all properties i)-iv).

Suppose now that a complete play $A_0 :: \ldots :: A_n$ of T_A has been played by Eloise following the above strategy and suppose by contradiction that $A_n = \texttt{False}$. Then, since by iv) Q_n is ω-correct and ω is winning, we have

$$\omega(Q_n) = A_0 :: \ldots :: A_i$$

for some $i \leq n$, which represents a backtracking move performed by ω. Since by ii) Q_n extends Q_i, we have that $Q_n, A_0 :: \ldots :: A_i$ can be written as

$$Q_i, Q, A_0 :: \ldots :: A_i$$

for some Q. As a consequence, $Q_i = Q'_i, A_0 :: \ldots A_i$ can be improved, which contradicts the optimality of Q_i stated at point iii).

Thus we have a winning strategy for Eloise, recursive in the oracles X_E, Φ_E. Now, however, we want Eloise to follow the very same strategy, but using *approximations* of those oracles in the place of the original ones. Of course, responses of approximated oracles are not to be always trustable. However, we will prove that correct oracle values can be learned by counterexamples and therefore that the use of the oracles may be replaced by a learning mechanism. According to learning based realizability, we will have in particular to prove that whenever the "approximated" strategy does not lead Eloise to win T_A, then some new value of the oracles can be learned: at least, from a failure Eloise corrects something old and gains something new, which is a perfect example of a *self-correcting* strategy.

Now, suppose that a complete play $A_0 :: \ldots :: A_n$ of T_A has been played by Eloise following the new "approximated" strategy. We still may suppose that $Q_0, \ldots, Q_n$ satisfy i), ii), iv). However, they satisfy only the following weaker

iii)$'$ For all i, Q_i is optimal, *according* to the response of the current oracle *approximations*.

In other words, it might happen that our approximated oracles believe Q_i to be not improvable, whilst Q_i actually *is*. Since iii) does not hold any more, the previous argument - that has shown $A_n = \texttt{True}$ - now fails and it might still occur that $A_n = \texttt{False}$. How Eloise is to learn from this counterexample? She computes $\omega(Q_n)$, which is of the form $A_0 :: \ldots :: A_i$, since it represents a backtracking move performed by ω. Then as before she writes $Q_n, A_0 :: \ldots :: A_i$ as

$$Q'_i, A_0 :: \ldots :: A_i, Q, A_0 :: \ldots :: A_i$$

As a consequence, she finds out that $Q'_i, A_0 :: \ldots A_i$ can be extended as to contradict the approximated-oracle prediction of point iii)$'$ and hence collects a new value of the oracle.

In the next two sections, we spell out the details of the construction and prove that the above Eloise strategy is sound and in fact convergent. In order to enhance readability and separate the important ideas from technicalities, we split the construction into two parts. First, we define the concept of learning strategy, which represents a translation of our realizability notion into the language of game theory, and show that any winning strategy in $\mathsf{1back}(T_A)$ can be transformed into a learning strategy in T_A. Secondly, we show that in fact any learning strategy in T_A can be translated into a learning based realizer of A.

4.6.2. Winning 1-Backtracking Strategies into Learning Strategies. For the rest of the paper, fix a closed implication-and-negation-free arithmetical formula A and let T_A be its associated Tarski game. Fix moreover a primitive recursive enumeration of the plays of $\mathsf{1back}(T_A)$ and let ω be a winning recursive strategy from A for player one in the game $\mathsf{1back}(T_A)$. We assume, without loss of generality, that ω performs backtracking moves only in front of atomic formulas; that is, we assume that for every play $Q, A_0 :: \ldots :: A_n$, if

$$\omega(Q, A_0 :: \ldots :: A_n) = A_0 :: \ldots :: A_i$$

then A_n is atomic. Clearly, any winning strategy can be transformed accordingly to this requirement: any backtracking move can be delayed by dummy moves and be performed in front of an atomic formula.

First of all, we formalize the coding of plays into numbers which has to be represented in our calculus.

Definition 4.6.2 (Abstract Plays of T_A, Coding terms). A sequence of arithmetical formulas $A_0 :: \ldots :: A_n$ is said to be an *abstract play* of T_A, if $A_0 = A$ and for all i

i) if $A_i = \forall x B$ or $A_i = \exists x B$, then $A_{i+1} = B$;

ii) if $A_i = B_0 \wedge B_1$ or $A_i = B_0 \vee B_1$, then $A_{i+1} = B_0$ or $A_{i+1} = B_1$.

Let moreover $p = A_0 :: \ldots :: A_k$ be any abstract play of T_A. By $|p| : \mathtt{N}$, we denote a term of $\mathcal{PCF}_{\text{Class}}$ having as free variables precisely the variables occuring free in the formulas of p and such that, for every sequence of numerals $\vec{n}$ and sequence of variables $\vec{x}$ comprising all the free variables of p, $|p|[\vec{n}/\vec{x}]$ is equal to the numeric code of the play $q = A_0[\vec{n}/\vec{x}] :: \ldots :: A_k[\vec{n}/\vec{x}]$.

□

We define now a translation of learning based realizability into the language of Tarski games. The translation consists of a pair (g_0, g_1) of terms of $\mathcal{PCF}_{\text{Class}}$: the first one describes a strategy for Eloise in T_A, and the second one decides when Eloise should backtrack.

Definition 4.6.3 (Learning Strategy). Let $T_A = (V, E_1, E_2, W)$. Let $g = (g_0, g_1)$ be a pair of terms of $\mathcal{PCF}_{\text{Class}}$ respectively of types $\mathtt{N} \to \mathtt{N}$ and $\mathtt{N} \to \mathtt{S}$.
For every state s, we say that a play $A_0 :: \ldots :: A_n$ of T_A is $g[s]$-*correct* if for every $i < n$ such that $(A_i, A_{i+1}) \in E_1$

$$g_0[s](|A_0 :: \ldots :: A_i|) = |A_{i+1}|$$

holds.
g is said to be a *learning strategy* from A if it satisfies the following conditions:

(1) (Soundness) For every state s and play $p = A :: A_0 :: \ldots :: A_n$ such that $A_n \in Dom(E_1)$, if $g_0[s](|p|) = |A_{n+1}|$, then $A :: A_0 :: \ldots :: A_n :: A_{n+1}$ is a play.

(2) (Convergence) For every play p starting from A, $g_0(|p|)$ and $g_1(|p|)$ converge.

(3) (Learning) For every state s and complete $g[s]$-correct play p starting from A, if

$$g_1[s](|p|) = \varnothing \implies p \in W$$

We observe that conditions (2) and (3) of definition 4.6.3 correspond respectively to the convergence property that $\mathcal{PCF}_{\text{Class}}$ realizers must have and to the learning condition in realizability for atomic formulas.

We now define a predicate $E : \mathtt{N}^2 \to \mathtt{Bool}$ of T, which codes the improvement relation between plays of $\mathsf{1back}(T_A)$ we are interested in. In the following, our terms of $\mathcal{PCF}_{\text{Class}}$ will make use only of the oracle constants X_E and Φ_E, which are in the syntax of $\mathcal{T}_{\text{Class}}$ (recall chapter 3, definition 3.2.6) and hence of $\mathcal{PCF}_{\text{Class}}$. Moreover, we define a term $\Psi : \mathtt{N} \to \mathtt{N}$, which will be our fundamental computational engine. Given a code of a ω-correct play, Ψ

is intended to return the code itself or the code of an optimal improvement, according to the oracles X_E, Φ_E. Ψ is in general non computable and by using Ψ as it is, we could only write strategies recursive in the oracles X_E, Φ_E, as in [9]. Therefore, we shall compute its approximations $\Psi[s]$, by which we will be able to write the "approximated" strategy for Eloise we have discussed in section 4.6.1.

Definition 4.6.4 (Improvement Relation, Optimality Operator Ψ). Let $E : \mathtt{N}^2 \to \mathtt{Bool}$ a predicate of Gödel's T such that $Enm = \mathtt{True}$ iff n and m are numerals coding respectively ω-correct plays $P, p :: A$ and $P, p :: A, Q, p :: A$ of $\mathsf{1back}(T_A)$, with $A = \exists x B(x)$ or $A = B_0 \vee B_1$.
We want to define now a term $\Psi : \mathtt{N} \to \mathtt{N}$ of $\mathcal{PCF}_{\text{Class}}$ such that the following equation is provable in the equational theory of $\mathcal{PCF}_{\text{Class}}$:

$$\Psi z = \mathsf{if}\ \mathsf{X}_E z\ \mathsf{then}\ \Psi(\Phi_E z)\ \mathsf{else}\ z$$

In order to do that, it is enough to let

$$\alpha := \lambda y^{\mathtt{N}\to\mathtt{N}} \lambda z^{\mathtt{N}} \mathsf{if}\ \mathsf{X}_E z\ \mathsf{then}\ y(\Phi_E z)\ \mathsf{else}\ z$$

and take $\Psi := \mathsf{Y}(\alpha)$.

The term Ψ, given a number n, checks whether $\mathsf{X}_E n = \mathtt{True}$, i.e. whether there exists a number m such that $Enm = \mathtt{True}$. In that case, it computes such an m by calling $\Phi_E n$, and continues the computation by calling itself on m; otherwise, it returns n. The termination of Ψn is guaranteed by the fact that there are no infinite ω-correct plays and each recursive call made by Ψ extends a current ω-correct play (see [9]). If X_E, Φ_E are interpreted as oracles, Ψn returns an optimal improvement of the play coded by n. But when X_E, Φ_E are approximated through a particular state s, in general $\Psi n[s]$ will return only an improvement of the play coded by n, or even n itself.
We now have to prove the crucial property that for any finite approximation of the oracles X_E, Φ_E - that is, for any state s - the term $\Psi n[s]$ has a normal form and that Ψn converge: Ψ is "stable" with respect to oracle approximations.

Proposition 4.6.1 (Stability of Ψ). $\Psi \in \|\mathtt{N} \to \mathtt{N}\|$

Proof. Let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of states. By definition 4.5.3, we have to prove that, for every term $t \in \|\mathtt{N}\|$, Ψt converges. Since t converges to a numeral, it is enough to show that for every numeral n, Ψn converges. First of all, we observe that for any state s, $\chi_E s m$ is equal to $\mathtt{True}$ only for a finite number of arguments m. Moreover, by definition 4.6.4

$$\Psi n[s] = \mathsf{if}\ \chi_E s n\ \mathsf{then}\ \Psi(\varphi_E s n)\ \mathsf{else}\ n$$

Hence, by direct computation it can be seen that, for every $i \in \mathbb{N}$, $\Psi n[s_i]$ has a normal form and it is equal, for some $k \in \mathbb{N}$, to $(\varphi_E s_i)^k n$, having defined by induction

$$(\varphi_E s_i)^0 n := n,\ (\varphi_E s_i)^{m+1} n := \varphi_E s_i((\varphi_E s_i)^m n)$$

Moreover, for every $m < k$

$$\chi_E s_i((\varphi_E s_i)^m n) = \mathtt{True} \tag{4.1}$$

does hold and hence $(\varphi_E s_i)^{m+1}n$ codes a play properly extending the play coded by $(\varphi_E s_i)^m n$. Now, if Ψn did not converge, then there would be two increasing infinite sequences of numbers $k_0, k_1, k_2, \ldots$ and $m_0, m_1, m_2, \ldots$ such that, for every $i \in \mathbb{N}$, $\Psi n[s_{m_i}] = (\varphi_E s_{m_i})^{k_i} n$. Furthermore, since $s_{m_i} \leq s_{m_{i+1}}$ and, by (4.1), for every $m < k_i$

$$\langle E, (\varphi_E s_{m_i})^m n, \varphi_E s_{m_i}((\varphi_E s_{m_i})^m n)\rangle \in s_{m_i}$$

we have that for every $m \leq k_i$

$$(\varphi_E s_{m_{i+1}})^m n = (\varphi_E s_{m_i})^m n$$

as it can be seen by induction on m. Hence, for every i, letting $a = k_{i+1} - k_i$

$$(\varphi_E s_{m_{i+1}})^{k_{i+1}} n = (\varphi_E s_{m+1})^a((\varphi_E s_{m_{i+1}})^{k_i} n) = (\varphi_E s_{m+1})^a((\varphi_E s_{m_i})^{k_i} n)$$

would be the code of a ω-correct play of $\mathsf{1back}(T_A)$ properly extending the play coded by $(\varphi_E s_{m_i})^{k_i} n$. Therefore, it would exist an infinite ω-correct play of $\mathsf{1back}(T_A)$, which is impossible since ω is a winning strategy by hypothesis.

□

We are going to define three terms Λ, Π, Ω that will implement the learning strategy for Eloise in T_A sketched in section 4.6.1. For the sake of readability, we will describe only the properties that such terms must satisfy, without explicitly write down those terms. It is trivial, however, to actually code our definitions in $\mathcal{PCF}_{\text{Class}}$.

We start by defining a term $\Lambda : \mathtt{N} \to \mathtt{S}$, which is supposed to code the function g_1 of definition 4.6.3. $\Lambda[s]$ takes the code of a play of $\mathsf{1back}(T_A)$ and builds a state containing values of the oracles X_E and $\mathsf{\Phi}_E$ that can be drawn from the input play and are not already in s.

Definition 4.6.5 (Learning Term). Let $\Lambda : \mathtt{N} \to \mathtt{S}$ be a term of $\mathcal{PCF}_{\text{Class}}$ described as follows. Λ takes as argument a numeral m. Then, it checks whether m codes a ω-correct play $Q = P, A_0 :: \ldots :: A_n$ of $\mathsf{1back}(T_A)$, with $A_0 :: \ldots :: A_n$ complete play of T_A and $A_n = \mathtt{False}$. If not, it returns a dummy state: $\varnothing$. Otherwise, it computes $\omega(Q) = A_0 :: \ldots :: A_i$, and, for any state s, returns $\Lambda[s]m$, which equals the state containing all the triples $\langle E, n_0, n_1\rangle$ not belonging to s and such that

i) n_0 codes a play $Q_0, A_0 :: \ldots :: A_i$;

ii) n_1 codes a play $Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i$;

iii) $Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i = Q, A_0 :: \ldots :: A_i$.

Note. The term Λ must make use of the constant Add_E to create its output $\Lambda[s]m$, which is a state, and that is why the triples $\langle E, n_0, n_1\rangle$ of the above definition 4.6.5 are not in s.

Recall subsection 4.6.1. We have explained that Eloise, while playing in T_A and in position $A_0 :: \ldots :: A_i$, simultaneously constructs a sequence of plays $Q_0, \ldots, Q_i$ satisfying some properties i)-iv). We now define a term $\Pi : \mathtt{N} \to \mathtt{N}$ of $\mathcal{PCF}_{\text{Class}}$, which will be used to construct that sequence. In particular, Π takes the code of a play $p = A_0 :: \ldots :: A_i$ and yields the code of a play $Q_i = Q, p$ of $\mathsf{1back}(T_A)$. Again, if Π is interpreted as a

program recursive in the oracles X_E, Φ_E, it will yield an optimal play Q, p. But when $\Pi[s]$ is computed, the result Q, p may not be optimal because X_E, Φ_E have been approximated through the state s.

Definition 4.6.6 (Sequence Constructor Π). We describe the behaviour of a term $\Pi : \mathtt{N} \to \mathtt{N}$ of $\mathcal{PCF}_{\text{Class}}$, intended to take the code $|p|$ of a play p of T_A and return the code $|Q, p|$ of a play Q, p of $\mathsf{1back}(T_A)$. The definition of $\Pi[s](|p|)$ runs by induction over the length of p and is distinguished by cases:

(1) If $p = A$, then $\Pi[s](|p|) = \Psi[s](|A|)$.

(2) If $p = (q :: B)$ and
$$\Pi[s](|q|) = |Q, q|$$
then
$$\Pi[s](|p|) = \Psi[s](|Q, q, q :: B|)$$

□

We now prove the convergence of Λ and Π.

Proposition 4.6.2 (Stability of Λ and Π). *$\Lambda \in \|\mathtt{N} \to \mathtt{S}\|$ and $\Pi \in \|\mathtt{N} \to \mathtt{N}\|$.*

Proof. Again, to prove that $\Lambda \in \|\mathtt{N} \to S\|$ it is enough to show that for every numeral n, Λn converges. Let then $\{s_i\}_{i \in \mathbb{N}}$ be a w.i. chain of states. By definition 4.6.5, whatever s_i is, $\Lambda[s_i]n$ construct a finite set of triples $\langle E, n_0, n_1 \rangle$, which depends only on n, and then decides to output some of the triples, according as to whether they are in s_i or not. This determination stabilizes for large enough s_m; that is, for all $m' \geq m$, $\Lambda[s_{m'}]n = \Lambda[s_m]$.

The convergence of Πn follows by straightforward induction on the length of the play coded by n and by the convergence of Ψ.

□

We now put together the terms Ψ, Λ, Π in order to define a learning strategy for Eloise in T_A.

Definition 4.6.7 (Learning Strategy for T_A). We describe the behavior of a pair of terms $\Omega = (\Omega_0, \Omega_1)$ respectively of type $\mathtt{N} \to \mathtt{N}$ and $\mathtt{N} \to \mathtt{S}$ of $\mathcal{PCF}_{\text{Class}}$, intended to represent a learning strategy for T_A. The definition of $\Omega_i[s](|p|)$ is distinguished by cases:

(1) If $p = q :: \exists x B(x)$ and
$$\Pi[s](|q :: \exists x B(x)|) = |Q, q :: \exists x B(x)|$$
and
$$\omega(Q, q :: \exists x B(x)) = q :: \exists x B(x) :: B(n)$$
then $\Omega_0[s](|p|) = |B(n)|$.

(2) If $p = q :: B_0 \vee B_1$ and
$$\Pi[s](|q :: B_0 \vee B_1|) = |Q, q :: B_0 \vee B_1|$$

and

$$\omega(Q, q :: B_0 \vee B_1) = q :: B_0 \vee B_1 :: B_i$$

then $\Omega_0[s](|p|) = |B_i|$.

(3) If $p = q :: B$, with B atomic, and

$$\Pi[s](|q :: B|) = |Q, q :: B|$$

then $\Omega_1[s](|p|) = \Lambda[s](|Q, q :: B|)$.

(4) In all other (trivial) cases, $\Omega_i[s](|p|)$, for $i = 0, 1$, can be arbitrarily defined as 0 or `True`.

PROPOSITION 4.6.3 (STABILITY OF Ω_0 AND Ω_1). *$\Omega_0 \in \|\mathtt{N} \to \mathtt{N}\|$ and $\Omega_1 \in \|\mathtt{N} \to \mathtt{S}\|$.*

PROOF. Trivial, by proposition 4.6.2.

□

We need first prove that the $\mathsf{1back}(T_A)$ plays constructed by Π are ω-correct, when Π starts from Ω[s]-correct plays of T_A built by Ω (this property correspond to property iv) of section 4.6.1).

LEMMA 4.6.1 . *Suppose $p := A_0 :: \ldots :: A_n$ is a $\Omega[s]$-correct play of T_A. Then*

$$\Pi[s](|A_0 :: \ldots :: A_n|) = |Q, A_0 :: \ldots :: A_n|$$

and $Q, A_0 :: \ldots :: A_n$ is ω-correct.

PROOF. Routine induction on n. If $n = 0$, then $p = A_0$. By definition of Π and Ψ

$$\Pi[s](|A_0|) = \Psi[s](|A_0|) = |Q, A_0|$$

with Q, A_0 ω-correct by construction of Ψ.
Suppose now $n > 0$. By induction hypothesis

$$\Pi[s](|A_0 :: \ldots :: A_{n-1}|) = |Q, A_0 :: \ldots :: A_{n-1}| \tag{4.2}$$

and $Q, A_0 :: \ldots :: A_{n-1}$ is ω-correct. If $A_{n-1} = \exists x B$ or $A_{n-1} = B \vee C$, then by definition of Ω, by equation (4.2) and Ω[s]-correctness of $A_0 :: \ldots :: A_n$, we have that

$$\Omega_0[s](|A_0 :: \ldots :: A_{n-1}|) = |A_n|$$

with

$$\omega(Q, A_0 :: \ldots :: A_{n-1}) = A_0 :: \ldots :: A_n$$

By definition 4.6.6 and equation (4.2)

$$\Pi[s](|A_0 :: \ldots :: A_n|) = \Psi[s](|Q, A_0 :: \ldots :: A_{n-1}, A_0 :: \ldots :: A_n|)$$

which is ω-correct. If $A_{n-1} = B \wedge C$ or $A_{n-1} = \forall x B$, then

$$\Psi[s](|Q, A_0 :: \ldots :: A_{n-1}, A_0 :: \ldots :: A_n|)$$

is automatically ω-correct.

□

We now prove the main theorem of this section: any recursive winning strategy ω for Eloise in $\mathsf{1back}(T_A)$ can be translated into a learning strategy from A for Eloise in T_A.

THEOREM 4.6.4 (1-BACKTRACKING STRATEGIES INTO LEARNING STRATEGIES). *Ω is a learning strategy for T_A.*

PROOF. The fact that Ω satisfies properties 1 and 2 of definition 4.6.3 is trivial and follows from proposition 4.6.3. So we prove property 3. Let s be a state and assume $p = A_0 :: \ldots :: A_n$ is a complete $\Omega[s]$-correct play of T_A. Suppose that $A_0 :: \ldots :: A_n \notin W$ and hence $A_n = \texttt{False}$. We have to prove that $\Omega_1[s]|p| \neq \varnothing$. By definition 4.6.7 of Ω, we have

$$\Omega_1[s]|p| = \Lambda[s](|Q, A_0 :: \ldots :: A_n|)$$

with

$$\Pi[s](|A_0 :: \ldots :: A_n|) = |Q, A_0 :: \ldots :: A_n|$$

By lemma 4.6.1 $|Q, A_0 :: \ldots :: A_n|$ is ω-correct, since $A_0 :: \ldots :: A_n$ is $\Omega[s]$-correct. By definition 4.6.5 of Λ, $\Omega_1[s]|p|$ contains all triples

$$\langle E, |Q_0, A_0 :: \ldots :: A_i|, |Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i| \rangle$$

not in s such that

$$\omega(Q, A_0 :: \ldots :: A_n) = A_0 :: \ldots :: A_i$$

and

$$Q, A_0 :: \ldots :: A_n, A_0 :: \ldots :: A_i = Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i$$

As implied by very definition 4.6.6 of Π, for every $j < n$, $\Pi[s](|A_0 :: \ldots :: A_{j+1}|)$ codes a play extending the play coded by $\Pi[s](|A_0 :: \ldots :: A_j|)$. Furthermore, for some Q', Q''

$$\Pi[s](|A_0 :: \ldots :: A_i|) = \Psi[s]|Q', A_0 :: \ldots :: A_i| = |Q'', A_0 :: \ldots :: A_i|$$

So, there is some Q''' such that

$$Q'', A_0 :: \ldots :: A_i, Q''' = Q, A_0 :: \ldots :: A_n$$

and most importantly, by definition of $\Psi[s]|Q', A_0 :: \ldots :: A_i|$

$$\chi_E s|Q'', A_0 :: \ldots :: A_i| = \texttt{False}$$

Therefore the triple

$$\langle E, |Q'', A_0 :: \ldots :: A_i|, |Q'', A_0 :: \ldots :: A_i, Q''', A_0 :: \ldots :: A_i| \rangle$$

belongs to $\Omega_1[s]|p|$, since it is not in s, by definition of $\mathsf{X}_E s$.

□

4.6.3. Learning Strategies into Realizers. In this section, we prove that the learning strategy Ω for T_A can be translated into a learning based $\mathcal{PCF}_{\text{Class}}$ realizer of A, thus proving our main completeness theorem.

We begin with a bit of coding.

DEFINITION 4.6.8 . Let $\Omega_0^{\mathtt{N}} : \mathtt{N} \to \mathtt{N}$ and $\Omega_0^{B} : \mathtt{N} \to \mathtt{Bool}$ be terms of $\mathcal{PCF}_{\text{Class}}$ such that:

(1) for every play $p :: \exists x B$ of T_A

$$\Omega_0^{\mathtt{N}}|p :: \exists x B| = n \iff \Omega_0|p :: \exists x B| = |B(n)|$$

(2) for every play $p :: B_0 \vee B_1$

$$\Omega_0^B|p :: B_0 \vee B_1| = \mathtt{True} \iff \Omega_0|p :: B_0 \vee B_1| = |B_0|$$

□

As a special case of the following definition, we get a candidate realizer for A.

DEFINITION 4.6.9 (REALIZER FOR A). Let p be an abstract play of T_A. We define by induction and by cases a term t_p of $\mathcal{PCF}_{\text{Class}}$, with free variables among those occurring free in some formula of p, as follows:

(1) $p = q :: \forall x B$.

$$t_{q::\forall x B} = \lambda x\ t_{q::\forall x B::B}$$

(2) $p = q :: \exists x B$.

$$t_{q::\exists x B} = \langle \Omega_0^{\mathtt{N}}|q :: \exists x B|, t_{q::\exists x B::B}[t_1/x]\rangle$$

where $t_1 := \Omega_0^{\mathtt{N}}|q :: \exists x B|$.

(3) $p = B_0 \vee B_1$.

$$t_{q::B_0\vee B_1} = \langle \Omega_0^B|q :: B_0 \vee B_1|, t_{q::B_0\vee B_1::B_0}, t_{q::B_0\vee B_1::B_1}\rangle$$

(4) $p = B_0 \wedge B_1$.

$$t_{q::B_0\wedge B_1} = \langle t_{q::B_0\wedge B_1::B_0}, t_{q::B_0\wedge B_1::B_1}\rangle$$

(5) $p = q$, with q complete.

$$t_q = \Omega_1|q|$$

□

LEMMA 4.6.2 (COMPLETENESS LEMMA). (1) *Let $\vec{m} = m_0, \ldots, m_k$ a sequence of closed stable type-$\mathtt{N}$ terms of $\mathcal{PCF}_{Class}$ and $\vec{x} = x_0, \ldots, x_k$ a sequence of variables containing all the free variables of $p :: B$. Then*

$$t_{p::B}[\vec{m}/\vec{x}] \in \|[B]\|$$

(2) *Let s be a state, $\vec{m} = m_0, \ldots, m_k$ a sequence of closed type-$\mathtt{N}$ terms of $\mathcal{PCF}_{Class}$, $\vec{x} = x_0, \ldots, x_k$ a sequence of variables and $(p :: B)[\vec{m}[s]/\vec{x}]$ a Ω[s]-correct play of T_A. Then*

$$t_{p::B}[\vec{m}/\vec{x}] \Vdash_s B[\vec{m}[s]/\vec{x}]$$

PROOF. We prove (1) by induction on p and by cases. We treat only three representative cases, those left out being obvious.

(1) $B = \forall x C$. Let n be a numeral. By inductive hypothesis, we have that

$$t_{p::\forall x C}[\vec{m}/\vec{x}]n = (\lambda x\ t_{p::\forall x C::C}[\vec{m}/\vec{x}])n$$

$$= t_{p::\forall x C::C}[\vec{m}, n/\vec{x}, x] \in \|[C]\|$$

Since $[\forall x C] = \mathtt{N} \to [C]$, we have that

$$t_{p::\forall x C}[\vec{m}/\vec{x}] \in \|[\forall x C]\|$$

(2) $B = \exists x C$. Let

$$t_1 := \Omega_0^{\mathtt{N}} | p :: \exists x C | [\vec{m}/\vec{x}]$$

Since the terms in $\vec{m}$ are stable by hypothesis and Ω_0 is stable by proposition 4.6.3 and by definition 4.6.8 of $\Omega_0^{\mathtt{N}}$, we have that t_1 is a stable term of type $\mathtt{N}$. By inductive hypothesis

$$t_{p::\exists x C::C}[\vec{m}, t_1/\vec{x}, x] \in \|[C]\|$$

Since $[\exists x C] = \mathtt{N} \times [C]$ and

$$t_{p::\exists x C}[\vec{m}/\vec{x}] = \langle \Omega_0^{\mathtt{N}} | p :: \exists x C | [\vec{m}/\vec{x}], t_{p::\exists x C::C}[\vec{m}, t_1/\vec{x}, x] \rangle$$

we have

$$t_{p::\exists x C}[\vec{m}/\vec{x}] \in \|[\exists x C]\|$$

(3) B atomic. Then

$$t_{p::B}[\vec{m}/\vec{x}] = \Omega_1 | p :: B | [\vec{m}/\vec{x}]$$

Since $[B] = \mathtt{S}$ and Ω_1 is stable by proposition 4.6.3, we have that $t_{p::B}[\vec{m}/\vec{x}] \in \|[B]\|$.

We now prove (2) by induction on p and by cases.

(1) $B = \forall x C$. Let n be a numeral. By inductive hypothesis, we have that

$$t_{p::\forall x C}[\vec{m}/\vec{x}]n = (\lambda x\ t_{p::\forall x C::C}[\vec{m}/\vec{x}])n$$
$$= t_{p::\forall x C::C}[\vec{m}, n/\vec{x}, x] \Vdash_s C[\vec{m}, n[s]/\vec{x}, x]$$

and hence

$$t_{p::\forall x C}[\vec{m}/\vec{x}] \Vdash_s \forall x C[\vec{m}[s]/\vec{x}]$$

(2) $B = \exists x C$. Suppose

$$t_1[s] := \Omega_0^{\mathtt{N}}[s] | p :: \exists x C | [\vec{m}[s]/\vec{x}] = n$$

with n numeral. By definition 4.6.8 of $\Omega_0^{\mathtt{N}}$, we have that

$$\Omega_0[s] | q :: \exists x C | [\vec{m}[s]/\vec{x}] = |C[\vec{m}[s], n/\vec{x}, x]|$$

and so $p :: \exists x C :: C[\vec{m}[s], n/\vec{x}, x]$ is $\Omega[s]$-correct. By inductive hypothesis

$$t_{p::\exists x C::C}[\vec{m}, t_1/\vec{x}, x] \Vdash_s C[\vec{m}[s], n/\vec{x}, x]$$

Since

$$t_{p::\exists x C}[\vec{m}/\vec{x}] = \langle \Omega_0^{\mathtt{N}} | p :: \exists x C | [\vec{m}/\vec{x}], t_{p::\exists x C::C}[\vec{m}, t_1/\vec{x}, x] \rangle$$

we have

$$t_{p::\exists x C}[\vec{m}/\vec{x}] \Vdash_s \exists x C[\vec{m}[s]/\vec{x}]$$

(3) $B = C_0 \wedge C_1$. By inductive hypothesis

$$t_{p::C_0 \wedge C_1::C_0}[\vec{m}/\vec{x}] \Vdash_s C_0[\vec{m}[s]/\vec{x}]$$

and

$$t_{p::C_0 \wedge C_1::C_1}[\vec{m}/\vec{x}] \Vdash_s C_1[\vec{m}[s]/\vec{x}]$$

Since

$$t_{p::C_0 \wedge C_1}[\vec{m}/\vec{x}] = \langle t_{p::C_0 \wedge C_1::C_0}[\vec{m}/\vec{x}], t_{p::C_0 \wedge C_1::C_1}[\vec{m}/\vec{x}] \rangle$$

we have

$$t_{p::C_0 \wedge C_1}[\vec{m}/\vec{x}] \Vdash_s C_0 \wedge C_1[\vec{m}[s]/\vec{x}]$$

(4) $B = C_0 \vee C_1$. Suppose

$$t_1[s] := \Omega_0^B[s]|p :: C_0 \vee C_1|[\vec{m}[s]/\vec{x}] = \texttt{True}$$

Then, by definition 4.6.8 of Ω_0^B, we have that

$$\Omega_0[s]|q :: C_0 \vee C_1|[\vec{m}[s]/\vec{x}] = |C_0[\vec{m}[s]/\vec{x}]|$$

and hence $p :: C_0 \vee C_1 :: C_0[\vec{m}[s]/\vec{x}]$ is $\Omega[s]$-correct. By inductive hypothesis

$$t_{p::C_0\vee C_1::C_0}[\vec{m}/\vec{x}] \Vdash_s C_0[\vec{m}[s]/\vec{x}]$$

Analogously, for $t_1[s] = \texttt{False}$, we have

$$t_{p::C_0\vee C_1::C_1}[\vec{m}/\vec{x}] \Vdash_s C_1[\vec{m}[s]/\vec{x}]$$

Since

$$t_{p::C_0\vee C_1}[\vec{m}/\vec{x}] = \langle \Omega_0^B|p :: C_0 \vee C_1|[\vec{m}/\vec{x}], t_{p::C_0\vee C_1::C_0}[\vec{m}/\vec{x}], t_{p::C_0\vee C_1::C_1}[\vec{m}/\vec{x}]\rangle$$

we have

$$t_{p::C_0\vee C_1}[\vec{m}/\vec{x}] \Vdash_s C_0 \vee C_1[\vec{m}[s]/\vec{x}]$$

(5) B atomic. Then

$$t_{p::B}[\vec{m}/\vec{x}][s] = \Omega_1[s]|p :: B|[\vec{m}[s]/\vec{x}]$$

Since $p :: B[\vec{m}[s]/\vec{x}]$ is $\Omega[s]$-correct and Ω is a learning strategy by lemma 4.6.2, if $t_{p::B[\vec{m}/\vec{x}]}[s] = \varnothing$, then $B[\vec{m}[s]/\vec{x}] = \texttt{True}$.

□

THEOREM 4.6.5 (COMPLETENESS THEOREM). *Suppose there exists a recursive winning strategy for player one in* $\mathsf{1back}(T_A)$. *Then there exists a term* t *of* $\mathcal{PCF}_{Class}$ *such that* $t \Vdash A$.

PROOF. By Lemma 4.6.2, point 1 and 2, applied to t_A and the empty sequence of terms. □

4.7. Conclusions

We have proved a soundness and completeness result for total recursive learning based realizability with respect to 1-Backtracking game semantics, solving a conjecture left open in Aschieri [4].

The contribution of the soundness theorem is semantical, rather than technical, and it should be useful to understand the significance and see possible uses of learning based realizability. We have shown how learning based realizers may be understood in terms of backtracking games and that this interpretation offers a way of eliciting constructive information from them. The idea is that playing games represents a way of challenging realizers; they react to the challenge by learning from failure and counterexamples. In the context of games, it is also possible to appreciate the notion of convergence, i.e. the fact that realizers stabilize their behaviour as they increase their knowledge. Indeed, it looks like similar ideas are useful to understand other classical realizabilities (see for example, Miquel [37]).

The proof of the completeness theorem has been definitely more technically challenging. In our view, moreover, it has two interesting features. In a sense, it is the first application

of the ideas of learning based realizability to a concrete non trivial classical proof, which is our version of the one given by Berardi et al. [9]. This proof classically shows that if Eloise has recursive winning strategy in the 1-Backtracking Tarski game associated to a formula A, then she also has a winning strategy in the Tarski game associated to A (but a strategy only recursive in an oracle for the Halting problem). Since the existence of this latter strategy implies the truth of A, the argument can be seen as a proof of A in some version of intuitionistic Arithmetic with EM_1. We managed to associate a constructive content to this seemingly ineffective proof and found out that it hides a learning mechanism to gain correct oracle values from failures and counterexamples. We have then transformed this learning mechanism into a learning based realizer of A. Secondly, we have shown the interesting theoretical result that backtracking strategies in 1-Backtracking games can interpreted as learning realizers. We have thus successfully established a close non trivial relationship between two interpretations of classical proofs: game semantics and learning based realizability.

CHAPTER 5

Constructive Analysis of Learning in Peano Arithmetic

ABSTRACT. In this chaper we give a constructive analysis of learning as it arises in various computational interpretations of classical Peano Arithmetic, such as our learning based realizability, Avigad's update procedures and epsilon substitution method. In particular, we show how to compute in Gödel's system T upper bounds on the length of learning processes, which are themselves represented in T through learning based realizability. The result is achieved by the introduction of a new non standard model of Gödel's T, whose new basic objects are pairs of non standard natural numbers (convergent sequences of natural numbers) and moduli of convergence, where the latter are objects giving constructive information about the former. As foundational corollary, we obtain that learning based realizability is a constructive interpretation of Heyting Arithmetic plus excluded middle over Σ^0_1 formulas (for which it was designed) and of all Peano Arithmetic when combined with Gödel's double negation translation. As byproduct of our approach, we also obtain a new proof of Avigad's theorem for update procedures and thus of termination of epsilon substitution method for PA.

5.1. Introduction

The aim of this chapter is to carry out a detailed and complete constructive analysis of learning, as it arises in learning based realizability for $\mathsf{HA}+\mathsf{EM}_1$ and in Avigad's [5] axiomatization of the epsilon substitution method for Peano Arithmetic through the concept of update procedure. The importance of this analysis is both practical and foundational. In the first place, we explicitly show how to compute upper bounds to the length of learning processes, thus providing the technology needed to analyze their computational complexity. Secondly, we answer positively to the foundational question of whether learning based realizability can be seen as an interpretation of classical Arithmetic into intuitionistic Arithmetic.

Our constructive framework is Gödel's system T and our metatheory will be purely intuitionistic. Our analysis will be accomplished by restating and then reproving constructively the following convergence theorem.

THEOREM 5.1.1 (CONVERGENCE). *Let $t : (\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$ be a closed term of T. Let $s : \mathtt{N} \to (\mathtt{N} \to \mathtt{N})$ be any closed term of T representing a weakly increasing chain of functions: that is, assume that for every numerals $n \leq m$, $s_n \leq s_m$[1] holds. Then, there exists an n such that for all $m \geq n$, $t(s_n) = t(s_m)$.*

The intuitive meaning of the convergence theorem is the following. It is intended to be an analysis of oracle computations. That is, given a non computable function $f : \mathtt{N} \to \mathtt{N}$

[1]Define $s_n \leq s_m$ iff for all numerals l, $s_n(l) \neq 0$ implies $s_n(l) = s_m(l)$. See the premise to definition 5.2.3 for intuitive meaning.

one would like to "compute" $t(f)$. Since this is not effectively possible, in order to obtain significant results one may try nevertheless to define a weakly increasing chain s of functions with the property that for all numerals n, $s_n \leq f$. Such a chain can be seen as a sequence of more and more refined approximations of f and can for example be constructed by means of *learning processes* as they arise in learning based realizability or epsilon substitution method (see Mints [35]). The theorem says that if t is computed with respect to such a sequence of approximations, then a stable answer about the value of $t(f)$ is eventually obtained.

The convergence theorem is already interesting in itself, but its special significance lies in its consequences, which we now describe and shall prove in the final part of the chapter. Since most of them cannot be proved if the convergence theorem is not first restated and then proven constructively, they provide an important motivation for working in this direction.

A first consequence of the convergence theorem is that any learning process represented by a learning based realizer always terminates. Formally:

THEOREM 5.1.2 (ZERO THEOREM). *Let t be a type $\mathtt{S}$ term of $\mathcal{T}_{Class}$. Define $s_0 := \varnothing$ and, for every natural number n, $s_{n+1} := s_n \Cup t[s_n]$. Then, there is an n such that $t[s_n] = \varnothing$.*

If the convergence theorem is proven constructively, also the above Zero theorem can be and so one obtains a constructive analysis of the numbers of learning steps required to complete the learning process. It has as a constructive consequence the following theorem:

THEOREM 5.1.3 (PROGRAM EXTRACTION VIA LEARNING BASED REALIZABILITY). *Let t be a term of $\mathcal{T}_{Class}$ and suppose that $t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$, with Pxy atomic. Then, from t one can define a term u of Gödel's system T such that for every numeral n, $Pn(un) = \mathtt{True}$.*

The above theorem sharpens the result obtained in chapter 3 and in Aschieri and Berardi [3]. There, we have proved as well that from any t such that $t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$ one can extract a computable function v such that for every numeral n, $Pn(vn) = \mathtt{True}$. However, the extracted v made use of unbounded iteration, while the u of theorem 5.1.3 is a "bounded" algorithm, that is, a program not explicitly using any kind of unbounded iteration. This is an important point from a foundational point of view: the algorithms extracted via learning based realizability *construct* witnesses, rather than *searching* for them. Let us make clear that, however, u - from the computational point of view - is equal to v. In fact, u results from v just by replacing its only unbounded iteration with a primitive recursive one (of an appropriate type). Thus, u just adds to v information about the computational complexity of the learning process generated by v. For practical purposes, therefore, v is as efficient as u.

As corollary, one obtains the important result that from classical proofs in Peano Arithmetic PA of $\forall\exists$-formulas one can extract bounded algorithms via learning based realizability $\Vdash$. This is done by, first, extracting a realizer from any given proof and, then, by applying theorem 5.1.3. In other words, one is able to give a novel proof of the following theorem due to Gödel (through its Dialectica interpretation, see e.g. [32]):

THEOREM 5.1.4 (PROVABLY TOTAL FUNCTIONS OF PA). *If $\mathsf{PA} \vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$, then there exists a term u of Gödel's system T such that for every numeral n, $Pn(un) = \mathtt{True}$.*

The novelty, here, is the technique employed to prove the theorem and the new understanding of extracted programs as realizers able to learn in a constructive way.

From a constructive proof of the convergence theorem one can also provide new constructive proofs of Avigad's [5] *fixed point theorem* for n-ary update procedures and hence of the *termination of the epsilon substitution method* for PA. Hence, one also obtains a *constructive analysis of learning* in Peano Arithmetic. The novelty, here, is the use of type theory to reason about the learning processes generated by update procedures and hence epsilon substitution method.

Theorem 5.1.1 can be proven easily, but ineffectively, in second order logic:

Proof of theorem 5.1.1 (Ineffective). The informal idea of the proof is the following. Terms of system T use only a finite number of values of their function arguments. If we "apply" t to the least upper bound f_s of the sequence s (w.r.t the relation $\leq$ of definition 5.2.3), we find that the finite part of f_s effectively used in the computation of $t(f_s)$ is already contained in some s_k. So, for every $h \geq k$, $t(s_h) = t(s_k)$.
Let us see the details. As proven by Kreisel (for a proof see Schwichtenberg [41]), t has a modulus of continuity $\mathcal{C}$, which is a term of system T of type $(\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$ such that the following statement is provable in extensional HA^ω:

$$\forall f^{\mathtt{N}\to\mathtt{N}}, g^{\mathtt{N}\to\mathtt{N}}.(\forall x^{\mathtt{N}} \leq (\mathcal{C}f)\ f(x) = g(x)) \to t(f) = t(g) \tag{5.1}$$

By using the comprehension axiom, we can define the least upper bound f_s of the sequence s as follows

$$f_s(n) = \begin{cases} m & \text{if } \exists i \text{ such that } s_i(n) = m \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{C}^M$ be the denotation of $\mathcal{C}$ in the full set theoretic model M of extensional HA^ω (see Kohlenbach [32]). Then there exists an n such that for all $m \geq n$

$$\forall x^{\mathtt{N}} \leq (\mathcal{C}^M f_s)\ s_n(x) = s_m(x)$$

By 5.1, we get that for all $m \geq n$, $t(s_n)^M = t(s_m)^M$. Hence by soundness of the model with respect to formal equality of extensional HA^ω, $t(s_n)$ and $t(s_m)$ normalize to the same numeral, since $t(s_n) = a$ and $t(s_m) = b$, with a, b numerals, implies $a^M = t(s_n)^M = t(s_m)^M = b^M$ and then $a = b$.

□

The convergence theorem is therefore true, but one cannot hope to prove it constructively as it is stated. In fact, it is a formula of the form $\forall\exists\forall$ and the simplest incompleteness of intuitionistic reasoning as compared to classical reasoning arises precisely for that kind of formulas. It is known, for example, that classical finite type Peano Arithmetic PA^ω proves the formula $\forall f^{\mathtt{N}\to\mathtt{N}}\exists x^{\mathtt{N}}\forall y^{\mathtt{N}} f(x) \leq f(y)$, while intuitionistic Heyting Arithmetic HA^ω does not. In our case, one could associate to any Turing machine a weakly increasing sequence $s : \mathtt{N} \to (\mathtt{N} \to \mathtt{N})$ such that for all m, $s_m(n) = 0$ if $n \neq 0$, and $s_m(0) = 1$ if the machine terminates on input n in less than m steps, $s_m(0) = 0$ otherwise. A constructive proof of the convergence theorem relatively to the term $\lambda f^{\mathtt{N}\to\mathtt{N}} f(0)$ would compute the limit of the sequence $\lambda m^{\mathtt{N}} s_m(0)$, thus determining whether the Turing machine terminates on input n. By producing such a sequence s for every Turing machine, we would have a solution for the Halting problem.

Synopsis of the chapter. In the rest of the chapter, we develop a technology for constructively reasoning about convergence in Gödel's system T and proving a classically equivalent form of the convergence theorem. All proofs will be constructive and all their constructive content will be made explicit. This constructive effort results in a longer and a bit more complex presentation than it could be. However, if one is not interested in full explicit details, the techniques used may be simplified in order to yield quite short and powerful constructive proofs of the main results of the chapter.

Our approach has a semantical content. In fact, we starts from considering a kind of constructive non standard model for Peano Arithmetic and then we reinterpret Gödel's system T constants in order to manipulate the new individuals of the model. The reinterpretation of system T will turn out to be particularly suited to perform the computations we need to do for constructively reasoning about convergence. From the high level point of view, the proof techiques used amount to a combination of Kreisel's no-counterexample interpretation and Tait's reducibility/logical-relations method. With the first one, we can constructively reason about convergence. With the second, we prove the soundness of the model with respect to our purposes.

In detail, the plan of the chapter is the following.

In section §5.2, we recall details of Gödel system T.

In section §5.3 we define the first ingredient of our approach, which is a constructive notion of convergence for sequences of objects, due to Berardi [6]. It is a no-counterexample interpretation of the classical notion of convergence, but it is different from the usual interpretation. Its main advantage is that it is very efficient from the computational point of view, since it enables *programming with continuations* and hence the writing of powerful and elegant realizers of its constructive content, which we will call *moduli of convergence*. Intuitively, a modulus of convergence for a convergent function $f : \mathtt{N} \to A$ will be a term able to find suitable intervals in which f is constant; moreover, the length of those intervals will depend on a continuation. At the end of the section we use Berardi's notion of convergence to reformulate the convergence theorem (see theorem 5.3.1).

In section §5.4, we introduce the second ingredient of our approach: a model that extends the usual full finite type structure generated over natural numbers by replacing naturals by pairs $\langle \mathcal{N}, f\rangle$ of a non standard natural number f (which is a function $\mathtt{N} \to \mathtt{N}$ as in ultrapower models of Peano Arithmetic) and its modulus of convergence $\mathcal{N}$. We also syntactically define a semantics $[\![\_]\!]_s$ (where s is a weakly increasing chain of functions) mapping terms of T in to elements of the model and in section §5.5 we show that, thanks to $[\![\_]\!]_s$, we can evaluate every term $t : (\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$, into a pair $\langle \mathcal{N}, f\rangle$ such that $\mathcal{N}$ is a modulus of convergence for the function $f = \lambda n^{\mathtt{N}} t(s_n)$.

In section §5.6, we prove all the corollaries of the convergence theorem that we have discussed before.

5.2. Term Calculus

In this chapter we will prove results that hold for any "simple" extension of Gödel's system T (see chapter 2). In this section, we recall the definition and results we shall need and introduce some useful notation.

Notation. For notational convenience and to define in a more readable way terms of type $A \times B \to C$, for any variables $x_0 : A$ and $x_1 : B$ we define

$$\lambda\langle x_0, x_1\rangle^{A\times B} u := \lambda x^{A\times B} u[\pi_0 x/x_0\ \pi_1 x/x_1]$$

where x is a fresh variable not appearing in u. We observe that for any terms t_0, t_1

$$(\lambda\langle x_0, x_1\rangle^{A\times B} u)\langle t_0, t_1\rangle = u[t_0/x_0\ t_1/x_0]$$

Often, as in chapter 3, it is useful to add to system T new constants and atomic types, together with a set of algebraic reduction rules we call "functional".

Definition 5.2.1 (Functional set of rules). Let C be any set of constants, each one of some type $A_1 \to \ldots \to A_n \to A$, for some atomic types $A_1, \ldots, A_n, A$. We say that $\mathcal{R}$ is a *functional set of reduction rules* for C if $\mathcal{R}$ consists, for all $c \in C$ and all closed normal terms $a_1 : A_1, \ldots, a_n : A_n$ of $\mathcal{T}$, of one and exactly one rule $ca_1 \ldots a_n \mapsto a$, where $a : A$ is a closed normal term of $\mathcal{T}$.

If a system $\mathcal{T}$ is obtained from Gödel's T by adding a recursive set C of constants and a recursive functional set of rules for C, we we call $\mathcal{T}$ a *simple extension of* T. As in chapter 3, by a standard reducibility argument it can be proved that $\mathcal{T}$ is strongly normalizing and has Church-Rosser property. Moreover, any atomic-type term of any simple extension $\mathcal{T}$ of T is equal either to a numeral, if it is of type $\mathtt{N}$, or to a boolean, if it is of type $\mathtt{Bool}$, or to a constant of type A, if it is of type A. All results of this paper hold whatever simple extension of T is chosen. Let us fix one.

Definition 5.2.2 (System $\mathcal{T}$). From now on, we denote with $\mathcal{T}$ an arbitrarily chosen simple extension of Gödel's system T. We also assume that $\mathcal{T}$ contains constants for deciding equality of constants of atomic type.

Throughout the paper, the intended interpretation of the natural number 0 will be as a "default" value. That is, when we do not have any information about what value a function has on argument n, we assume that it has value 0. That being said, it is natural to consider a function $f_1 : \mathbb{N} \to \mathbb{N}$ to be *extending* another function $f_2 : \mathbb{N} \to \mathbb{N}$, whenever it holds that for every n such that $f_1(n)$ is a non default value (and hence different from 0), then $f_1(n) = f_2(n)$. f_2 may hence have a non default value at some argument where f_1 has a default value, but it agrees with f_1 at the arguments where f_1 has not default value. So, f_2 carries more information than f_1.

Definition 5.2.3 (Ordering Between Functions and Terms). Let f_1, f_2 be functions $\mathbb{N} \to \mathbb{N}$. We define

$$f_1 \leq f_2 \iff \forall n \in \mathbb{N}\ f_1(n) \neq 0 \Rightarrow f_1(n) = f_2(n)$$

Moreover, if t_1, t_2 are closed terms of $\mathcal{T}$ of type $\mathtt{N} \to \mathtt{N}$ representing respectively functions $g_1, g_2 : \mathbb{N} \to \mathbb{N}$, we will write $t_1 \leq t_2$ if and only if $g_1 \leq g_2$.

In the following, we will write "$s \in$ w.i." if is $s : \mathtt{N} \to (\mathtt{N} \to \mathtt{N})$ is a closed term representing a weakly increasing sequence of functions, that is, if for all numerals n, m, $n \leq m$ implies $s_n \leq s_m$.

5.3. The No-Counterexample Interpretation and Berardi's Notion of Convergence

In this paper, we are interested in arithmetical formulas stating convergence of natural number sequences. Classically, we consider a sequence of natural numbers to be convergent if it is definitely constant, that is, if the there is an element of the sequence which is equal to all successive elements of the sequence. Hence, we will consider formulas of the form

$$(\forall z^A)\ \exists x^{\mathtt{N}} \forall y^{\mathtt{N}} P(z, x, y) \tag{5.2}$$

Since that kind of formulas cannot generally be proven constructively, a common standpoint is to consider classically equivalent but constructively weak enough statements, as in Kreisel *no-counterexample interpretation*:

$$(\forall z^A)\ \forall f^{\mathtt{N}\to\mathtt{N}} \exists x^{\mathtt{N}} P(z, x, f(x))$$

If the statement 5.2 (with $A = \mathtt{N}$) is provable in PA, then one can constructively extract from any proof a term $t : (\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$ of system $\mathcal{T}$ such that

$$(\forall z^{\mathtt{N}})\ \forall f^{\mathtt{N}\to\mathtt{N}} P(z, t(f), f(t(f)))$$

holds (see for example Kohlenbach [32]). In our cases, we have to deal with formulas of the form

$$\exists x^{\mathtt{N}} \forall y^{\mathtt{N}} \geq x f(x) = f(y)$$

where f is a term of type $\mathtt{N} \to \mathtt{N}$, and hence we may be tempted to consider their no-counterexample interpretation

$$\forall h^{\mathtt{N}\to\mathtt{N}} \exists x^{\mathtt{N}}\ h(x) \geq x \to f(x) = f(h(x)) \tag{5.3}$$

If one introduces the notation

$$f \downarrow [n, m] \stackrel{\text{def}}{\equiv} \forall x^{\mathtt{N}}.\ n \leq x \leq m \to f(x) = f(n)$$

one often finds in literature the following equivalent version of 5.3:

$$\forall h^{\mathtt{N}\to\mathtt{N}} \exists x^{\mathtt{N}}\ f \downarrow [x, h(x)] \tag{5.4}$$

which is the no-counterexample interpretation of

$$\exists x^{\mathtt{N}} \forall y^{\mathtt{N}} \geq x f \downarrow [x, y]$$

While the above notion of convergence 5.4 would be enough for our purposes, it seems not to allow straightforward compositional reasoning when one has to deal with non trivial *interaction* of convergent functions. Even when there is no complex interaction, the needed reasoning is not direct. For example, one may want to prove that if two functions f, g converge in the sense of 5.4, one can systematically find intervals in which they are *both* constant. That is, if

$$\forall h^{\mathtt{N}\to\mathtt{N}} \exists x^{\mathtt{N}} f \downarrow [x, h(x)] \wedge \forall h^{\mathtt{N}\to\mathtt{N}} \exists x^{\mathtt{N}} g \downarrow [x, h(x)]$$

then one may want to prove that

$$\forall h^{\mathtt{N}\to\mathtt{N}}\exists x^{\mathtt{N}} f\downarrow[x,h(x)]\wedge g\downarrow[x,h(x)]$$

The above implication is provable in a non overly complicated way, but when interaction increases (as we shall see in proposition 5.3.3 below), one begins to feel the need for a more suitable formulation of convergence.

Berardi [6] introduced a notion of convergence especially suited for managing interaction of convergent functions. If one consider the formula

$$\forall z^{\mathtt{N}}\exists x^{\mathtt{N}}\geq z\forall y^{\mathtt{N}}\geq x f\downarrow[x,y]$$

(with the intent of expressing very redundantly the fact that there are infinite points of convergence for f) one obtains a very strong notion of constructive convergence by taking its no-counterexample interpretation

$$\forall z^{\mathtt{N}}\forall h^{\mathtt{N}\to\mathtt{N}}\exists x^{\mathtt{N}}\geq z\ h(x)\geq x\to f\downarrow[x,h(x)]$$

which after skolemization becomes

$$\forall h^{\mathtt{N}\to\mathtt{N}}\exists\alpha^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}\forall z^{\mathtt{N}}\ h(z)\geq z\to f\downarrow[\alpha(z),h(\alpha(z))]$$

which is equivalent to

$$\forall h^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}\exists\alpha^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}\ \forall z^{\mathtt{N}} f\downarrow[\alpha(z),h(\alpha(z))] \tag{5.5}$$

where we have used the notation

$$\alpha^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}\stackrel{\text{def}}{\equiv}\forall x^{\mathtt{N}}\alpha(x)\geq x$$

We observe that 5.4 and 5.5 are constructively equivalent. However, from a computational point of view, their realizers are quite different: the realizers of 5.5 are able to interact directly with each other, as we will see.

We are now ready to formally define a constructive notion of convergence for sequences of numbers: a sequence of objects $f:\mathtt{N}\to A$ is convergent if for any $h^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}$ there are infinitely many intervals $[n,h(n)]$ in which f is constant.

Definition 5.3.1 (Convergence (Berardi [6])). Let $f:\mathtt{N}\to A$ be a closed term of $\mathcal{T}$, with A atomic type. We say that f *converges* if

$$\forall h^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}\ \exists\alpha^{\mathtt{N}\to\mathtt{N}}\geq\mathsf{id}\ \forall z^{\mathtt{N}} f\downarrow[\alpha(z),h(\alpha(z))]$$

Notation. If $t:A\to B$ and $u:A$ we shall often write t_u in place of tu, for notational convenience or for highlighting that t_u is an element of a collection of type-B terms parametrized by terms of type A.

We now make explicit the constructive information associated to the above notion of convergence, through the concept of *modulus of convergence*. A modulus of convergence takes an $h:\mathtt{N}\to\mathtt{N}$ and returns an enumeration of intervals $[n,h(n)]$ in which f is constant. It is a intuitionistic realizer of the notion of convergence.

DEFINITION 5.3.2 (MODULUS OF CONVERGENCE). Let $f : \mathtt{N} \to A$ be a closed term of $\mathcal{T}$, with A atomic type. A term $\mathcal{M} : (\mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})$ of $\mathcal{T}$ is a *modulus of convergence* for f if

(1) $\forall h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}\ \mathcal{M}_h \geq \mathsf{id}$

(2) $\forall h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}\ \forall z^{\mathtt{N}} f \downarrow [\mathcal{M}_h(z), h(\mathcal{M}_h(z))]$

If $h : \mathtt{N} \to \mathtt{N} \geq \mathsf{id}$ and $\forall z^{\mathtt{N}} f \downarrow [\mathcal{N}(z), h(\mathcal{N}(z))]$, $\mathcal{N}$ is said to be an *h-modulus of convergence* for f.

We observe that by definition, if one has a modulus of convergence $\mathcal{M}$ for a function f, one can find an infinite number of intervals of any desired length in which f is constant. For example, if one wants to find an interval of length 5, one just defines the function $h(x) = x + 5$ and compute $n := \mathcal{M}_h(0)$. Then, f is constant in $[n, n + 5]$. Clearly, a modulus of convergence carries a lot of constructive information about f.

5.3.1. Intuitive Significance of the Concept of Modulus of Convergence and Restatement of the Convergence Theorem. As we said, Berardi's notion of convergence works remarkably well when convergent functions interact together, for instance, in the definition of a new function. The fact that Berardi's notion is a no-counterexample interpretation of the classical notion of convergence, explains why it *works*. We can intuitively describe the reasons why it does it *well* as follows.

A first reason is purely computational. Given a function $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$ and a modulus of convergence $\mathcal{M}$, we can interpret the role of h in the computation of $\mathcal{M}_h$ as that of a *continuation*. Constructively, when a new convergent function is defined from other convergent functions, one will need to produce intervals in which the new function is constant. One will try to achieve the goal by finding intervals in which the functions involved in the definition are *all* constant. The problem is that one may be able to find such intervals for every single function, but not for them all together. For example, if one defines the function

$$\beta := \lambda x^{\mathtt{N}} f(g(x), x)$$

then β is convergent if g and $\lambda x^{\mathtt{N}} f(n, x)$ are such for every choice of n. But an interval in which g is constant need not be an interval in which β too is constant, because we have to find some interval in which *both* g is constantly equal to some m and $\lambda x^{\mathtt{N}} f(m, x)$ is constant. We solve the problem through the use of continuations.

We start by observing that it seems there is a strict sequence of tasks to be performed. First, one tries to find an m_1 such that g is constant in, say, $[m_1, l_1]$ with $m_1 < l_1$. Then, he computes $g(m_1) = n$ and pass n to a "continuation" $h : \mathtt{N} \to \mathtt{N}$ which returns an $h(n) = m_2 < l_2$ such that $\lambda x^{\mathtt{N}} f(n, x)$ is constant in $[m_2, l_2]$. If $m_1 < m_2 < l_1$, a non trivial interval in which g is constant has been found. But if $m_2 > l_1$? Then, g may assume different values in all points of the interval $[m_2, l_2]$ and one cannot hope that β is going to be convergent in $[m_2, l_2]$. We anticipate the solution contained in the proof of proposition 5.3.3, by letting $m_1 = \mathcal{M}_k(0)$, where $\mathcal{M}$ is a modulus of convergence for g and, for example, $k(x) = h(g(x)) + 1$. Then, by definition of modulus of convergence, g is constant in $[m_1, k(m_1)]$ and letting $l_1 = k(m_1)$ we obtain that

$$m_2 = h(n) = h(g(m_1)) < h(g(m_1)) + 1 = l_1$$

as required. In other words, we use k and hence h as *continuations*, thanks to $\mathcal{M}$.

The issue we are facing may be further exemplified by the following sequential game between k players. Suppose there are convergent functions $f_1, f_2, \ldots, f_k$ ot type $\mathtt{N} \to \mathtt{N}$ on the board and an arbitrarily chosen number m. Players make their moves in order, starting from player one and finishing with player k. A play of the game, is an increasing sequence of numbers $m, m_1, m_2, \ldots, m_k$, with m_i the move of player i. Player i wins if f_i is constant in an interval $[m_k, l_k]$, for some $l_k > m_k$. A strategy for player i is just a function h over natural numbers, taking the move of the player $i-1$ (or the integer m if $i = 1$) and returning the move of player i. The fact that the winning condition depends on the move of player k makes very difficult for players $1, \ldots k-1$ to win. In this game, each player hopes that in the resulting final interval its own function will be constant but his hope is frustrated by the following ones, which are trying to accomplish the same task but with respect to their own functions. However, player i has a winning strategy effectively computable if he knows the strategies of all subsequent players $i+1, \ldots, k$ (we cannot assume the trivial winning strategy returning the point of stabilization of f_i to be effectively computable, since f_i is arbitrary)

We are now in a position to tell another reason why moduli of convergence are so useful. A winning strategy for player i can be computed by a convergence module. More precisely, it can be proved, as consequence of proposition 5.3.2, that if players $i+1, \ldots, k$ play strategies $h_{i+1}, \ldots, h_k$, then $h_i := \mathcal{M}_{h_k \circ \cdots \circ h_{i+1}}$ is a winning strategy for player i against $h_{i+1}, \ldots, h_k$, whenever $\mathcal{M}$ is a modulus of convergence for f_i. Therefore, if a modulus of convergence for each function $f_1, \ldots, f_k$ is given, one can compute a particularly desirable instance of *Nash equilibrium*, that is, a sequence of functions $h_1, h_2, \ldots, h_k$ such that, if every player i plays according to the strategy h_i, every play will be won by every player. Therefore, at the end of the interaction, every participant will have accomplished its own task.

We now formulate the promised restatement of theorem 5.1.1 that we shall be able to prove.

THEOREM 5.3.1 (WEAK CONVERGENCE). *Let $t : (\mathtt{N} \to \mathtt{N}) \to \mathtt{S}$ be a closed term of $\mathcal{T}$, with $\mathtt{S}$ atomic type. Then we can effectively define a closed term $\mathcal{M} : (\mathtt{N} \to (\mathtt{N} \to \mathtt{N})) \to (\mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})$ of $\mathcal{T}$, such that the following holds: for all $s : \mathtt{N} \to (\mathtt{N} \to \mathtt{N})$, $s \in$ w.i. and numerals n, $\mathcal{M}s$ is a modulus of convergence for $\lambda m^{\mathtt{N}} t_n(s_m)$.*

5.3.2. Basic Operations with Moduli of Convergence. We now prove a couple of propositions, both to illustrate the use of moduli of convergence and to provide lemmas we will need in the following. First, we show that given two terms f_1 and f_2, if each one of them has a modulus of convergence, then there is a modulus of convergence that works simultaneously for both of them. In particular, we can define a binary operation $\sqcup$ between moduli of convergence such that, for every pair of moduli $\mathcal{M}, \mathcal{N}$, $\mathcal{M} \sqcup \mathcal{N}$ is "more general" than both $\mathcal{M}$ and $\mathcal{N}$. Here, for every $\mathcal{M}_1, \mathcal{M}_2$, we call $\mathcal{M}_2$ more general than $\mathcal{M}_1$, if for every term f, if $\mathcal{M}_1$ is a modulus of convergence for f then also $\mathcal{M}_2$ is a modulus of convergence for f. We this terminology, we may see $\mathcal{M} \sqcup \mathcal{N}$ as an upper bound of the set $\{\mathcal{M}, \mathcal{N}\}$, with respect to the partial order induced by the relation "to be more general than". The construction of the pair $\mathcal{M}_{h \circ \mathcal{N}_h}, \mathcal{N}_h$ below may also be seen as a Nash equilibrium for the two player version of the game we have discussed above.

PROPOSITION 5.3.2 (JOINT CONVERGENCE). *Let $\mathcal{M}$ and $\mathcal{N}$ be moduli of convergence respectively for f_1 and f_2. Define*

$$\mathcal{M} \sqcup \mathcal{N} := \lambda h^{\mathtt{N}\to\mathtt{N}} \lambda z^{\mathtt{N}} \mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z))$$

Then $\mathcal{M} \sqcup \mathcal{N}$ is a modulus of convergence for both f_1 and f_2.

PROOF. Set

$$\mathcal{L} := \mathcal{M} \sqcup \mathcal{N}$$

First, we check property 1 of definition 5.3.2. For all $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$, $\mathcal{N}_h \geq \mathsf{id}$ by definition 5.3.2 point (1) and so $h \circ \mathcal{N}_h \geq \mathsf{id}$. Thus, for all $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$ and $z^{\mathtt{N}}$

$$\mathcal{L}_h(z) = \mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z)) \geq z$$

since $\mathcal{M}$ has property (1) of definition 5.3.2 and hence $\mathcal{M}_{h\circ\mathcal{N}_h} \geq \mathsf{id}$. Therefore, for all $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$, $\mathcal{L}_h \geq \mathsf{id}$ and we are done.
Secondly, we check property 2 of definition 5.3.2. Fix a term $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$ and a numeral z. We have that

$$f_1 \downarrow [\mathcal{M}_{h\circ\mathcal{N}_h}(z), h \circ \mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z))] \tag{5.6}$$

since $\mathcal{M}$ is a module of convergence for f_1. Moreover,

$$f_2 \downarrow [\mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z)), h(\mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z)))] \tag{5.7}$$

since $\mathcal{N}$ is a modulus of convergence for f_2. But the starting point of the interval in 5.7 is greater or equal to the starting point of the interval in 5.6, for $\mathcal{N}_h \geq \mathsf{id}$, while their ending points are equal. Hence also

$$f_1 \downarrow [\mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z)), h(\mathcal{N}_h(\mathcal{M}_{h\circ\mathcal{N}_h}(z)))]$$

and hence both f_1 and f_2 are constant in the interval $[\mathcal{L}_h(z), h(\mathcal{L}_h(z))]$ by definition of $\mathcal{L}$. □

We now consider a situation in which a family $\{f_n\}_{n\in\mathtt{N}}$ of convergent terms interacts with a convergent term g and we show the result of the interaction is still a convergent term. In the following, we call "object of type A" any closed normal term of type A.

PROPOSITION 5.3.3 (MERGING OF FUNCTIONS). *Let $f : A \to (\mathtt{N} \to A)$ be a closed term, with A atomic, and $\mathcal{N} : A \to (\mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})$ be such that for every object a of type A, $\mathcal{N}_a$ is a modulus of convergence for f_a. Let moreover $g : \mathtt{N} \to A$ and let $\mathcal{M}$ be a modulus of convergence for g. Define*

$$\mathcal{H}_1(\mathcal{M}, \mathcal{N}, g) := \lambda h^{\mathtt{N}\to\mathtt{N}} \lambda z^{\mathtt{N}} \mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z))$$

with

$$\mathcal{N}'_h := \lambda n^{\mathtt{N}}(\mathcal{N}_{g(n)}h)n$$

Then $\mathcal{H}_1(\mathcal{M}, \mathcal{N}, g)$ is a modulus of convergence for

$$\lambda n^{\mathtt{N}} f_{g(n)}(n)$$

PROOF. Property (1) of definition 5.3.2 follows by the same reasoning used in proposition 5.3.2. We check property (2) of definition 5.3.2. Set $\mathcal{L} := \mathcal{H}_1(\mathcal{M}, \mathcal{N}, g)$. The idea is that $\mathcal{L}$ has to produce an interval i in which g is constant and equal to a, while the interval produced by $\mathcal{N}_a$ in which f_a is constant will be contained in i. $\mathcal{L}$ does the job by using $\mathcal{N}'_h$ as a *continuation*.
Fix a term closed $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$ and z a numeral. We have that

$$g \downarrow [\mathcal{M}_{h\circ\mathcal{N}'_h}(z), h \circ \mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z))] \tag{5.8}$$

since $\mathcal{M}$ is a module of convergence for g. In particular,

$$g \downarrow [\mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)), h(\mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)))] \tag{5.9}$$

since $\mathcal{N}'_h \geq \mathsf{id}$. Say that for all n in the intervals in 5.8 and 5.9, $g(n) = a$. By definition of $\mathcal{N}'_h$

$$[\mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)), h(\mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)))]$$

$$= [\mathcal{N}_a h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)), h(\mathcal{N}_a h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)))] \tag{5.10}$$

Since $\mathcal{N}_a$ is a modulus of convergence for f_a, we have

$$f_a \downarrow [\mathcal{N}_a h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)), h(\mathcal{N}_a h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)))]$$

But for all x in the interval 5.10,

$$(\lambda n^{\mathtt{N}} f_{g(n)}(n))x = f_a(x)$$

Hence

$$\lambda n^{\mathtt{N}} f_{g(n)}(n) \downarrow [\mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)), h(\mathcal{N}'_h(\mathcal{M}_{h\circ\mathcal{N}'_h}(z)))]$$

and so $\lambda n^{\mathtt{N}} f_{g(n)}(n)$ is constant in the interval $[\mathcal{L}_h(z), h(\mathcal{L}_h(z))]$ by definition of $\mathcal{L}$.

□

5.4. Computations with non Standard Natural Numbers

For technical convenience we add now to system $\mathcal{T}$ a constant $\Phi : \mathtt{N} \to \mathtt{N}$ with no associated reduction rules. In this way, each term $t : A$ can be viewed as functionally depending on Φ, but it is still considered as having type A, instead the more complicated $(\mathtt{N} \to \mathtt{N}) \to A$. Of course, terms of atomic type are not in general equal to a constant or a numeral, if they contain Φ.

DEFINITION 5.4.1 (EVALUATION AT u). Let t be a term. For any term $u : \mathtt{N} \to \mathtt{N}$, we denote with $t[u]$ the term $t[u/\Phi]$.

Adopting this notation, what we want prove is that if $t : A$, with A atomic, and $s \in$ w.i., then the function $\lambda m^{\mathtt{N}} t[s_m]$ constructively converges, that is, it has a modulus of convergence. A natural attempt for achieving the goal is to recursively decompose the problem. For example, suppose we want to study the convergence of the function

$$^*(+t_1t_2) := \lambda m^{\mathtt{N}} + t_1t_2[s_m] : \mathtt{N} \to \mathtt{N}$$

where $s \in$ w.i.. and $+ : \mathtt{N} \to \mathtt{N} \to \mathtt{N}$ represents a constant of $\mathcal{T}$ encoding the operation of addition of natural numbers. Since $t_1 : \mathtt{N}$ and $t_2 : \mathtt{N}$ may have complex structure, it is natural to recursively study the functions

$$^*t_1 := \lambda m^{\mathtt{N}} t_1[s_m] : \mathtt{N} \to \mathtt{N}$$

and

$$^*t_2 := \lambda m^{\mathtt{N}} t_2[s_m] : \mathtt{N} \to \mathtt{N}$$

But if we want to study the function $^*(+t_1t_2)$ as a combination of $^*t_1$ and $^*t_2$, it is clear that $+$ cannot be interpreted as itself, but as a function $^*+$ of $^*t_1$ and $^*t_2$. We would like the following equation to hold

$$^*(+t_1t_2) = {}^*+{}^*t_1{}^*t_2$$

As a consequence of our notation, also the following equation must be true for all numerals n

$$^*(+t_1t_2)(n) = +(^*t_1(n))(^*t_2(n))$$

These considerations impose us to define

$$^*+ := \lambda g_1^{\mathtt{N}\to\mathtt{N}} \lambda g_2^{\mathtt{N}\to\mathtt{N}} \lambda m^{\mathtt{N}} + g_1(n)g_2(n)$$

At a first look, this may seem a rather strange way of doing computations. But it turns out that it is *strongly* not the case. $^*t_1$ and $^*t_2$ may be interpreted as *hypernatural* numbers and $^*+$ as the operation of addition of hypernaturals as they are defined in ultrapower non standard models of Peano Arithmetic.

5.4.1. Non Standard Models of Arithmetic. The first non standard model of Arithmetic is due to Skolem [42]. The universe of that model is indeed made of functions $\mathbb{N} \to \mathbb{N}$, but we instead describe a variant of the Skolem construction, which is the ultrapower construction (see for example Goldblatt [23]).

Fix a non principal ultrafilter $\mathcal{F}$ over $\mathbb{N}$. First, define an equivalence relation $\simeq$ between functions $\mathbb{N} \to \mathbb{N}$ as follows:

$$f_1 \simeq f_2 \iff \{x \in \mathbb{N} \mid f_1(x) = f_2(x)\} \in \mathcal{F}$$

(The intuition here is that an ultrafilter collects the "big" subsets of $\mathbb{N}$ and hence two functions are to be considered equal if they have equal values for "great many" arguments. For example, two functions which, as sequences, converge to the same natural number are considered equal, for they agree on a cofinite set of $\mathbb{N}$, which must belong to every non principal ultrafilter). Secondly, define

$$^*\mathbb{N} := (\mathbb{N} \to \mathbb{N})_{\simeq}$$

that is, $^*\mathbb{N}$ is the set of all natural number functions partitioned under the equivalence relation $\simeq$. Finally, set

$$\begin{aligned}
^*0 &:= \lambda n^{\mathbb{N}} 0 \\
^*S &:= \lambda n^{\mathbb{N}} S(n) \\
^*+ &:= \lambda f_1^{\mathbb{N}\to\mathbb{N}} \lambda f_2^{\mathbb{N}\to\mathbb{N}} \lambda n^{\mathbb{N}} f_1(n) + f_2(n) \\
^*\cdot &:= \lambda f_1^{\mathbb{N}\to\mathbb{N}} \lambda f_2^{\mathbb{N}\to\mathbb{N}} \lambda n^{\mathbb{N}} f_1(n) \cdot f_2(n)
\end{aligned}$$

where $S, +, \cdot$ are the usual operations over natural numbers. In general, if one wants to define the non standard version of a standard function $f : \mathbb{N}^k \to \mathbb{N}$, he simply lets

$$^*f := \lambda f_1^{\mathbb{N}\to\mathbb{N}} \dots \lambda f_k^{\mathbb{N}\to\mathbb{N}} \lambda n^{\mathbb{N}} f(f_1(n), \dots, f_k(n))$$

It can be proved that the structure

$$({}^*\mathbb{N}, {}^*0, {}^*S, {}^*+, {}^*\cdot)$$

is a model of Peano Arithmetic as similar to the usual structure of natural numbers as to satisfy *precisely* the same sentences which are true under the usual interpretation. Formally, it is *elementarily equivalent* to the structure of natural numbers.

Elements of ${}^*\mathbb{N}$ are usually called hypernatural numbers. Since they are so similar to natural numbers, it perfectly makes sense to think about defining a model of system $\mathcal{T}$ over hypernaturals. Indeed, Berardi [7] used hypernaturals, under a weaker equivalence relation, to construct an intuitionistic model for Δ_0^2 maps and Berardi and de' Liguoro [12] used them to interpret a fragment of classical primitive recursive Arithmetic.

5.4.2. A non Standard Model for the System T_0. In order to approach gradually our final construction, we first give a definition of a non standard model for T_0, which is Gödel's T restricted to having only a recursion operator R of type $\mathtt{N} \to (\mathtt{N} \to \mathtt{N} \to \mathtt{N}) \to \mathtt{N} \to \mathtt{N}$ and choice operator $\mathsf{if} : \mathtt{Bool} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N}$. Hence, T_0 represents the primitive recursive functions.

The definition of the model is purely syntactical and this is the key for our approach to go through. In fact, we are defining an internal model, that is a representation of T_0 into T itself. First, define the new type structure as:

$$\begin{aligned} {}^*\mathtt{N} &:= \mathtt{N} \to \mathtt{N} \\ {}^*\mathtt{Bool} &:= \mathtt{N} \to \mathtt{Bool} \\ {}^*(A \to B) &:= {}^*A \to {}^*B \\ {}^*(A \times B) &:= {}^*A \times {}^*B \end{aligned}$$

From a semantical point of view, we interpret natural numbers as functions. Since the construction is syntactical, there is no need to describe an equivalence relation between those functions. But, accordingly to which equivalence relation one has in mind, the definition we are going to give will make sense or not from the *semantical* point of view. For the results of this chaper, we have no utility in putting extra effort to define a model for T_0, which is also a model for Peano Arithmetic. Hence, we may assume that ${}^*\mathtt{N}$ represents just all functions over $\mathbb{N}$ without any partition.

Now, for every term $u : T$, define a term ${}^*u$ of type ${}^*T$ by induction as follows

$$
\begin{aligned}
{}^*0 &:= \lambda m^{\mathtt{N}}\ 0\\
{}^*\mathtt{True} &:= \lambda m^{\mathtt{N}}\ \mathtt{True}\\
{}^*\mathtt{False} &:= \lambda m^{\mathtt{N}}\ \mathtt{False}\\
{}^*\mathsf{S} &:= \lambda f^{{}^*\mathtt{N}} \lambda m^{\mathtt{N}}\ \mathsf{S}(f(m))\\
{}^*\mathsf{if} &:= \lambda g^{{}^*\mathtt{Bool}} \lambda f_1^{{}^*\mathtt{N}} \lambda f_2^{{}^*\mathtt{N}} \lambda m^{\mathtt{N}}\ \mathsf{if} f(m) f_1(m) f_2(m)\\
{}^*\mathsf{R} &:= \lambda f_1^{{}^*\mathtt{N}} \lambda f_2^{{}^*\mathtt{N}\to{}^*\mathtt{N}\to{}^*\mathtt{N}} \lambda g^{{}^*\mathtt{N}} \lambda m^{\mathtt{N}}\ (\mathsf{R}_B f_1 (\lambda n^{\mathtt{N}} f_2 (\lambda x^{\mathtt{N}} n)) g(m))(m)\\
{}^*(x^A) &:= x^{{}^*A}\\
{}^*(ut) &:= {}^*u\,{}^*t\\
{}^*(\lambda x^A u) &:= \lambda x^{{}^*A}\,{}^*u\\
{}^*\langle u, t\rangle &:= \langle {}^*u, {}^*t\rangle\\
{}^*(\pi_i u) &:= \pi_i\,{}^*u
\end{aligned}
$$

with the type B of R_B equal to ${}^*\mathtt{N} \to (\mathtt{N} \to {}^*\mathtt{N} \to {}^*\mathtt{N}) \to \mathtt{N} \to {}^*\mathtt{N}$.

The definition of the constants and the functions ${}^*\mathsf{S}$ and ${}^*\mathsf{if}$ is exactly the one used in the construction of ultrapower models of natural numbers. The definition of ${}^*\mathsf{R}$ is different because involves higher type arguments, but it is a straightforward generalization of the ultrapower construction. Intuitively, ${}^*\mathsf{R} f_1 f_2 g$ has to iterate f_2 a number of times given by g. But since g is now an hypernatural number, the concept "g times" makes no direct sense. Hence, ${}^*\mathsf{R}$ also picks as input a number m, transform g into $g(m)$ and iterates f_2 a number of times given by $g(m)$. But since f_2 is of type ${}^*\mathtt{N} \to {}^*\mathtt{N} \to {}^*\mathtt{N}$, the function given to R_B is not directly f_2, but a term $\lambda n^{\mathtt{N}} f_2(\lambda x^{\mathtt{N}} n)$ that transforms n into its hypernatural counterpart and gives it as the first argument of f_2. After all this work is done, one obtains a hypernatural

$$h := \mathsf{R}_B f_1 (\lambda n^{\mathtt{N}} f_2 (\lambda x^{\mathtt{N}} n)) g(m)$$

So if ${}^*\mathsf{R}$ stopped here, it would not return the right type of object. Hence, it returns $h(m)$, consistently to the fact that g has been instantiated to m previously.

The above construction can be generalized to Gödel's T, with a little more effort to be put in the generalization of ${}^*\mathsf{R}$ and ${}^*\mathsf{if}$ to all types. A version of $\mathcal{T}$ just manipulating hypernaturals is not enough for our purposes, and will be included in our final construction, so details are postponed to the next sections.

5.4.3. System $\mathcal{T}$ over Hypernaturals with Moduli of Convergence. In the context of this work, we are not interested into the whole collection of hypernatural numbers, but only in those who are *convergent*. Moreover, we want also to produce, for each one of these convergent hypernaturals, a modulus of convergence. The idea therefore is to put more constructive information into the model of hypernatural numbers and to define operations that preserve this information. The new objects we are going to consider are *hypernatural numbers with moduli of convergence*. They can be represented as pairs

$$\langle \mathcal{N}, f\rangle$$

where $f : \mathbb{N} \to \mathbb{N}$ is an hypernatural number and $\mathcal{N} : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$ is modulus of convergence for f, as in definition 5.3.2. The resulting model will be a full type structure generated over this basic objects and their equivalent in the other atomic types. We will

call it the *model of hypernaturals with moduli*. In the following, for any $s \in$w.i., $[\![u]\!]_s$ will be the denotation of a term u of $\mathcal{T}$ in this new model and the aim of this sections is to syntactically define the interpretation function $[\![\_]\!]_s$.

In order to construct such a model, we will have to define new operations that, first, generalize the ones over hypernaturals we have previously studied and, secondly, are also able to combine moduli of convergence.

For example, how to define the non standard version $[\![+]\!]_s$ of addition? The summands are two objects of the form $\langle \mathcal{N}_1, f_1 \rangle$ and $\langle \mathcal{N}_2, f_2 \rangle$. The second component of the sum will be the non standard sum

$$f_1{}^* + f_2 := \lambda m^{\mathtt{N}} f_1(m) + f_2(m)$$

of f_1 and f_2. The first component will be a modulus of convergence for $f_1{}^* + f_2$, and so a simultaneous modulus of convergence for both f_1 and f_2 is enough. From proposition 5.3.2, we know how to compute it with $\sqcup$ from $\mathcal{N}_1$ and $\mathcal{N}_2$. We can thus define

$$[\![+]\!]_s \langle \mathcal{N}_1, f_1 \rangle \langle \mathcal{N}_2, f_2 \rangle := \langle \mathcal{N}_1 \sqcup \mathcal{N}_2, f_1{}^* + f_2 \rangle$$

We now launch into the definition of our syntactically described model for the whole system $\mathcal{T}$. First we define the intended interpretation M_T of every type T.

Definition 5.4.2 (Interpretation of Types). For every type T of system $\mathcal{T}$, we define a type M_T by induction on T as follows.

(1) $T = A$, with A atomic. Then

$$\mathsf{M}_A := ((\mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})) \times (\mathtt{N} \to A)$$

(2) $T = A \to B$. Then

$$\mathsf{M}_{A \to B} := \mathsf{M}_A \to \mathsf{M}_B$$

(3) $T = A \times B$. Then

$$\mathsf{M}_{A \times B} := \mathsf{M}_A \times \mathsf{M}_B$$

If A is atomic, the interpretation M_A of A is the set of pairs formed by a function $\mathbb{N} \to A$ and its modulus of convergence (if it happens to have one). This is an accord with our view that whenever a $s \in$w.i. is fixed, a term t of atomic type can be interpreted as a function $\lambda m^{\mathtt{N}} t[s_m]$ paired with a modulus of convergence. The model of hypernaturals with moduli can be seen as the collection of sets denoted by types M_T, for T varying on all types of $\mathcal{T}$.

We now define a logical relation between the terms of our intended model of hypernaturals with moduli and the terms of system $\mathcal{T}$. It formally states what properties any denotation of any term of $\mathcal{T}$ should have. It formalizes of our previous description of what the model should contain.

Definition 5.4.3 (Generalized Modulus of Convergence). Let t and $\mathcal{M}$ be closed terms of $\mathcal{T}$ and $s \in$w.i.. We define the relation $\mathcal{M}$ gmc_s t - representing the notion "$\mathcal{M}$ is a *generalized modulus of convergence* for t" - by induction on the type T of t as follows:

(1) $T = A$, with A atomic. Let $\mathcal{M} : \mathsf{M}_A$. Then

$$\mathcal{M}\ \mathsf{gmc}_s\ t \iff \mathcal{M} = \langle \mathcal{L}, g \rangle,\ \mathcal{L} \text{ is a modulus of convergence for } g \text{ and } g \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} t[s_n]$$

where we have defined $(g \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} t[s_n]) \equiv$ for all numerals m, $g(m) = t[s_m]$.

(2) $T = A \to B$. Let $\mathcal{M} : \mathsf{M}_{A\to B}$. Then

$$\mathcal{M}\ \mathsf{gmc}_s\ t \iff (\forall u^A.\ \mathcal{N}\ \mathsf{gmc}_s\ u \implies \mathcal{M}\mathcal{N}\ \mathsf{gmc}_s\ tu)$$

(3) $T = A \times B$. Let $\mathcal{M} : \mathsf{M}_{A\times B}$. Then

$$\mathcal{M}\ \mathsf{gmc}_s\ t \iff (\pi_0\mathcal{M}\ \mathsf{gmc}_s\ \pi_0 t \wedge \pi_1\mathcal{M}\ \mathsf{gmc}_s\ \pi_1 t)$$

The aim of the rest of this section is to syntactically define a semantic interpretation $[\![\_]\!]_s$ of the terms of $\mathcal{T}$ into the model of hypernaturals with moduli, such that for every term $u : A$ and $s \in$w.i., $[\![u]\!]_s\ \mathsf{gmc}_s\ u$. This means that, if A is atomic, u is evaluated in a pair $\langle \mathcal{L}, g\rangle$ such that $g \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} u[s_n]$ and $\mathcal{L}$ is a modulus of convergence of g. Then, if given any term $t : (\mathtt{N} \to \mathtt{N}) \to A$, we set $u := t\Phi$ and consider $[\![u]\!]_s$, we automatically obtain a constructive proof of theorem 5.3.1.

In the following, we will make repeated use of the fact that the notion of generalized modulus of convergence is consistent with respect to equality.

Lemma 5.4.1 (Equality Soundness). *Suppose* $\mathcal{M}_1\ \mathsf{gmc}_s\ t_1$, $\mathcal{M}_1 = \mathcal{M}_2$ *and* $t_1 = t_2$. *Then* $\mathcal{M}_2\ \mathsf{gmc}_s\ t_2$.

Proof. Trivial induction on the type of T. □

We now define a fundamental operation on moduli of convergence. The construction is a generalization of the one in proposition 5.3.3.

Definition 5.4.4 (Collection of Moduli Turned into a Single Modulus). Let $\mathcal{N} : A \to \mathsf{M}_T$ and $\langle \mathcal{M}, g\rangle : \mathsf{M}_A$, with A atomic. We define by induction on T and by cases a term $\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})$ of type M_T.

(1) T atomic. Then

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) := \langle \mathcal{H}_1(\mathcal{M}, \lambda a^A \pi_0(\mathcal{N}_a), g),\ \lambda n^{\mathtt{N}} f_{g(n)}(n)\rangle$$

with $f := \lambda a^A \pi_1 \mathcal{N}_a$ and $\mathcal{H}_1$ as in proposition 5.3.3.

(2) $T = C \to B$. Then

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) := \lambda \mathcal{L}^{\mathsf{M}_C} \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \mathcal{N}_a \mathcal{L})$$

(3) $T = C \times B$. Then

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) := \langle \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \pi_0 \mathcal{N}_a), \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \pi_1 \mathcal{N}_a)\rangle$$

If we call "object of type A" any closed normal term of type A, then the role of the term $\mathcal{H}$ is to satisfy the following lemma, which is one the most important pieces of our construction. It provides a way of constructing the semantics of a term ut, with t of atomic type A, if one is able to define a semantics for t and for ua for every object a of type A.

LEMMA 5.4.2 . *Let u and t be closed terms respectively of types $A \to T$ and A, with A atomic. Suppose that for every object a of type A, $\mathcal{N}_a$ gmc_s ua and $\langle \mathcal{M}, g\rangle$ gmc_s t. Then $\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})$ gmc_s ut.*

PROOF. By induction on T and by cases.

(1) T atomic. We have

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) := \langle \mathcal{H}_1(\mathcal{M}, \lambda a^A \pi_0(\mathcal{N}_a), g), \lambda n^{\mathtt{N}} f_{g(n)}(n)\rangle$$

with

$$f := \lambda a^A \pi_1 \mathcal{N}_a$$

and

$$g \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} t[s_n]$$

for by hypothesis $\langle \mathcal{M}, g\rangle$ gmc_s t. Moreover, for every object a of type A

$$f_a \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} ua[s_n]$$

since by hypothesis $\mathcal{N}_a$ gmc_s ua. We must show that

$$\mathcal{H}_1(\mathcal{M}, \lambda a^A \pi_0 \mathcal{N}_a, g)$$

is a modulus of convergence for the function $\lambda n^{\mathtt{N}} f_{g(n)}(n)$ and that $\lambda n^{\mathtt{N}} f_{g(n)} \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} ut[s_n]$. For this last part, indeed, for every numeral m, there is an object $a = g(m)$ such that

$$\begin{aligned}(\lambda n^{\mathtt{N}} f_{g(n)}(n))m &= f_a(m)\\ &\overset{\text{ext}}{=} u[s_m](a)\\ &= u[s_m](g(m))\\ &\overset{\text{ext}}{=} u[s_m]((\lambda n^{\mathtt{N}} t[s_n])m)\\ &= u[s_m](t[s_m])\\ &= (\lambda n^{\mathtt{N}} ut[s_n])m\end{aligned}$$

Now, since $\langle \mathcal{M}, g\rangle$ gmc_s t, $\mathcal{M}$ is a modulus of convergence for g. Moreover, for every object a of type A, $\mathcal{N}_a$ gmc_s ua by hypothesis, and therefore $\pi_0 \mathcal{N}_a$ is a modulus of convergence for $\lambda n^{\mathtt{N}}$ $ua[s_n] \overset{\text{ext}}{=} f_a$. By proposition 5.3.3, we obtain that

$$\mathcal{H}_1(\mathcal{M}, \lambda a^A \pi_0 \mathcal{N}_a, g)$$

is modulus of convergence for $\lambda n^{\mathtt{N}} f_{g(n)}(n)$, and we are done.

(2) $T = C \to B$. Let $v : C$ and suppose $\mathcal{L}$ gmc_s v. We have to show that

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})\mathcal{L} = \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \mathcal{N}_a \mathcal{L})\ \mathsf{gmc}_s\ utv$$

But for every object a of type A, $\mathcal{N}_a$ gmc_s ua. Therefore, for every object a of type A

$$\mathcal{N}_a \mathcal{L}\ \mathsf{gmc}_s\ uav = (\lambda m^A umv)a$$

By induction hypothesis

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \mathcal{N}_a \mathcal{L})\ \mathsf{gmc}_s\ (\lambda m^A umv)t = utv$$

which is the thesis.

(3) $T = C \times B$. We have to show that, for $i = 0, 1$,

$$\begin{aligned}\pi_i \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) &= \pi_i \langle \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \pi_0 \mathcal{N}_a), \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \pi_1 \mathcal{N}_a)\rangle \\ &= \mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \pi_i \mathcal{N}_a)\rangle \ \mathsf{gmc}_s \ \pi_i(ut)\end{aligned}$$

Now, for every object a of type A, $\mathcal{N}_a \ \mathsf{gmc}_s \ ua$. Therefore, for every object a of type A

$$\pi_i \mathcal{N}_a \ \mathsf{gmc}_s \ \pi_i(ua) = (\lambda m^A \pi_i(um))a$$

By induction hypothesis

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \lambda a^A \pi_i \mathcal{N}_a) \ \mathsf{gmc}_s \ (\lambda m^A \pi_i(um))t = \pi_i(ut)$$

which is the thesis.

□

We are now in a position to define for each constant c of $\mathcal{T}$ a term $[\![c]\!]_s$, which is intended to satisfy the relation $[\![c]\!]_s \ \mathsf{gmc}_s \ c$. $[\![c]\!]_s$ can be seen as the non standard version of the operation denoted by c.

Definition 5.4.5 (Generalized Moduli of Convergence for Constants). We define for every constant $c : T$ a closed term $[\![c]\!]_s : \mathsf{M}_T$, accordingly to the form of c.

(1) $c : A$, A atomic. For any closed term u of atomic type, define

$$\mathcal{M}_{\mathsf{id},u} := \langle \lambda h^{\mathtt{N}\to\mathtt{N}} \lambda m^{\mathtt{N}} m, \lambda n^{\mathtt{N}} u\rangle$$

Then

$$[\![c]\!]_s := \mathcal{M}_{\mathsf{id},c}$$

(2) $c = \Phi : \mathtt{N} \to \mathtt{N}$. Let

$$\mathcal{N} := \lambda n^{\mathtt{N}} \langle \lambda h^{\mathtt{N}\to\mathtt{N}} \lambda m^{\mathtt{N}} \mathsf{if} \ s_m(n) = s_{h(m)}(n) \ \mathsf{then} \ m \ \mathsf{else} \ h(m), \lambda m^{\mathtt{N}} \ s_m(n)\rangle$$

Then

$$[\![\Phi]\!]_s := \lambda \langle \mathcal{M}, g\rangle^{\mathsf{M}_{\mathtt{N}}} \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})$$

(3) $c \neq \Phi, c \neq \mathsf{if}$, $c : A_0 \to \cdots \to A_m \to A$, with A, A_i atomic for $i = 0, \ldots, m$.
If $m > 0$, then define

$$\begin{aligned}[\![c]\!]_s := \lambda \langle \mathcal{L}_0, g_0\rangle^{\mathsf{M}_{A_0}} \ldots \lambda \langle \mathcal{L}_m, g_m\rangle^{\mathsf{M}_{A_m}} \langle \mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \ldots \sqcup \mathcal{L}_m, \\ \lambda n^{\mathtt{N}} c(g_0(n)) \ldots (g_m(n))\rangle\end{aligned}$$

assuming left association for $\sqcup$.
If $m = 0$ ($c : A_0 \to A$), define

$$[\![c]\!]_s := \lambda \langle \mathcal{M}, g\rangle^{\mathsf{M}_{A_0}} \langle \mathcal{M}, \lambda n^{\mathtt{N}} c(g(n))\rangle$$

(4) $c = \mathsf{R}_T$, R_T recursor constant with $T = A \to (\mathtt{N} \to A \to A) \to \mathtt{N} \to A$. Define

$$\mathcal{N} := \lambda n^{\mathtt{N}}\ \mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n})n$$

with

$$U := \mathsf{M}_A \to (\mathtt{N} \to \mathsf{M}_A \to \mathsf{M}_A) \to \mathtt{N} \to M_A$$

Then

$$[\![\mathsf{R}_T]\!]_s := \lambda \mathcal{I}^{\mathsf{M}_A} \lambda \mathcal{L}^{\mathsf{M}_{\mathtt{N}\to A\to A}} \lambda \langle \mathcal{M}, g\rangle^{\mathsf{M}_{\mathtt{N}}} \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})$$

(5) $c = \mathsf{if}_T$ with $T : \mathtt{Bool} \to A \to A \to A$. Define

$$\mathcal{N} := \lambda b^{\mathtt{Bool}} \mathsf{if}\ b\ \mathsf{then}\ \mathcal{L}_1\ \mathsf{else}\ \mathcal{L}_2,$$

Then

$$[\![\mathsf{if}_T]\!]_s := \lambda \langle \mathcal{M}, g\rangle^{\mathsf{M}_{\mathtt{Bool}}} \lambda \mathcal{L}_1^{\mathsf{M}_A} \lambda \mathcal{L}_2^{\mathsf{M}_A} \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})$$

The definition of $[\![c]\!]_s$ is a generalization of the operations done with hypernaturals. In case (1), we transform basic objects into their hypernatural, hyperboolean and hyperconstant counterparts (we call them hyperobjects) all paired with their trivial moduli of convergence.

In case (2), the interpretation $[\![\Phi]\!]_s$ of Φ is obtained by first defining uniformly on the numeral parameter n a collection of interpretations $\mathcal{N}_n = [\![\Phi n]\!]_s$ of Φn, and then using the term $\mathcal{H}$ to put together the interpretations in such a way that $[\![\Phi]\!]_s\langle \mathcal{M}, g\rangle$ interprets $[\![\Phi t]\!]_s$ whenever $\langle \mathcal{M}, g\rangle = [\![t]\!]_s$.

In case (3), we provide the non standard version of the function represented by c, which is a function $[\![c]\!]_s$ which combines both hyperobjects and their moduli of convergence.

In case (4) and (5) we have generalized the ideas of subsection 5.4.2. In particular, for $T = A \to (\mathtt{N} \to A \to A) \to \mathtt{N} \to A$ and A atomic, the definition of $[\![\mathsf{R}_T]\!]_s$ is exactly the same of ${}^*\mathsf{R}$ in subsection 5.4.2, enriched with the information of how to combine moduli of convergence. In fact, if we consider the term

$$\begin{aligned}&[\![\mathsf{R}_T]\!]_s \mathcal{I}\mathcal{L}\langle \mathcal{M}, g\rangle\\ =&\langle \mathcal{H}_1(\mathcal{M}, \lambda n^{\mathtt{N}} \pi_0(\mathcal{N}_n), g),\ \lambda n^{\mathtt{N}} f_{g(n)}(n)\rangle\end{aligned}$$

with $f := \lambda n^{\mathtt{N}} \pi_1(\mathcal{N}_n)$, its right projection is equal to

$$\lambda n^{\mathtt{N}} f_{g(n)}(n)$$

which is equal to

$$\lambda n^{\mathtt{N}} (\pi_1 \mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n}) g(n))(n)$$

which correspond exactly to the term

$${}^*\mathsf{R} f_1 f_2 g = \lambda n^{\mathtt{N}}\ (\mathsf{R}_B f_1 (\lambda n^{\mathtt{N}} f_2(\lambda x^{\mathtt{N}} n)) g(n))(n)$$

of subsection 5.4.2.

We now prove that for any constant c, $[\![c]\!]_s$ is a generalized modulus of convergence for c.

PROPOSITION 5.4.1 . *For every constant c, $[\![c]\!]_s$ gmc_s c.*

PROOF. We proceed by cases, accordingly to the form of c.

(1) $c = \Phi$. Let $t : \mathtt{N}$ and suppose $\langle \mathcal{M}, g\rangle \ \mathsf{gmc}_s \ t$. We have to prove that

$$[\![\Phi]\!]_s\langle \mathcal{M}, g\rangle \ \mathsf{gmc}_s \ \Phi t$$

By definition 5.4.5 of $[\![\Phi]\!]_s$

$$[\![\Phi]\!]_s\langle \mathcal{M}, g\rangle = \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})$$

with

$$\mathcal{N} := \lambda n^{\mathtt{N}}\langle \lambda h^{\mathtt{N}\to\mathtt{N}}\lambda m^{\mathtt{N}}\mathsf{if}\ s_m(n) = s_{h(m)}(n)\ \mathsf{then}\ m\ \mathsf{else}\ h(m), \lambda m^{\mathtt{N}} s_m(n)\rangle$$

Since $\langle \mathcal{M}, g\rangle \ \mathsf{gmc}_s \ t$, if we prove that for every numeral n, $\mathcal{N}_n \ \mathsf{gmc}_s \ \Phi n$, we obtain by lemma 5.4.2 that $\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) \ \mathsf{gmc}_s \ \Phi t$ and we are done. So let us show that, given a numeral n, $\pi_0\mathcal{N}_n$ is a modulus of convergence for the function

$$\pi_1\mathcal{N}_n = \lambda m^{\mathtt{N}} s_m(n) \overset{\text{ext}}{=} \lambda m^{\mathtt{N}} \ \Phi(n)[s_m]$$

We have to prove that given any closed term $h^{\mathtt{N}\to\mathtt{N}} \geq \mathsf{id}$ and numeral n,

$$\lambda m^{\mathtt{N}} s_m(n) \downarrow [(\pi_0\mathcal{N}_n)_h(z), h((\pi_0\mathcal{N}_n)_h(z))]$$

We have two possibilities:
i) $s_z(n) = s_{h(z)}(n)$. Since $s \in$ w.i., we have either $s_{h(z)}(n) = 0$ and so

$$\forall y^{\mathtt{N}}.\ z \leq y \leq h(z) \implies s_y(n) = 0$$

or $s_z(n) = s_{h(z)}(n) \neq 0$ and so

$$\forall y^{\mathtt{N}}.\ z \leq y \leq h(z) \implies s_y(n) = s_z(n)$$

Therefore

$$\begin{aligned}\lambda m^{\mathtt{N}} s_m(n) \downarrow &[z, h(z)]\\ &= [(\pi_0\mathcal{N}_n)_h(z), h((\pi_0(\mathcal{N}_n)_h(z))]\end{aligned}$$

by definition of $\mathcal{N}$.
ii) $s_z(n) \neq s_{h(z)}(n)$. Since $s \in$ w.i., we have $s_z \leq s_{h(z)}$ and hence $0 = s_z(n)$. So $s_{h(z)}(n) = s_{h(h(z))}(n)$ and as above

$$\begin{aligned}\lambda m^{\mathtt{N}} s_m(n) \downarrow &[h(z), h(h(z))]\\ &= [(\pi_0\mathcal{N}_n)_h(z), h((\pi_0\mathcal{N}_n)_h(z))]\end{aligned}$$

(2) $c \neq \Phi$, $c \neq \mathsf{if}$, $c : A_0 \to \cdots \to A_m \to A$.
i) $m > 0$. Suppose $t_i : A_i$ and $\langle \mathcal{L}_i, g_i\rangle \ \mathsf{gmc}_s \ t_i$ for all $i = 0, \ldots, m$. We have to prove that

$$[\![c]\!]_s\langle \mathcal{L}_0, g_0\rangle \ldots \langle \mathcal{L}_m, g_m\rangle \ \mathsf{gmc}_s \ ct_1 \ldots t_m$$

We have that $g_i \overset{\text{ext}}{=} \lambda n^{\mathtt{N}} t_i[s_n]$ for $i = 0, \ldots, m$. Moreover, since by definition 5.4.5 of $[\![c]\!]_s$

$$\begin{aligned}&[\![c]\!]_s\langle \mathcal{L}_0, g_0\rangle \ldots \langle \mathcal{L}_m, g_m\rangle\\ =&\langle \mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \ldots \sqcup \mathcal{L}_m, \lambda n^{\mathtt{N}} c(g_0(n)) \ldots (g_m(n))\rangle\end{aligned}$$

we must show that

$$\begin{aligned}&\pi_0([\![c]\!]_s\langle \mathcal{L}_0, g_0\rangle \ldots \langle \mathcal{L}_m, g_m\rangle)\\ =&\mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \ldots \sqcup \mathcal{L}_m\end{aligned}$$

is a modulus of convergence for

$$\begin{aligned}&\lambda n^{\mathtt{N}} c(g_0(n))\dots(g_m(n))\\ \overset{\text{ext}}{=}&\lambda n^{\mathtt{N}} c(t_1[s_n])\dots(t_m[s_n])\\ =&\lambda n^{\mathtt{N}} c t_1\dots t_m[s_n]\end{aligned}$$

Since for $i=0,\dots,m$, $\mathcal{L}_i$ is a modulus of convergence for g_i, by repeated application of proposition 5.3.2 we deduce that $\mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \dots \sqcup \mathcal{L}_m$ is modulus of convergence for all $g_1,\dots,g_m$ simultaneously. Hence for all closed terms $h : \mathtt{N}\to\mathtt{N} \geq \mathsf{id}$ and numerals z, and for $i=0,\dots,m$

$$g_i \downarrow [(\mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \dots \sqcup \mathcal{L}_m)_h(z), h((\mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \dots \sqcup \mathcal{L}_m)_h(z))]$$

and therefore

$$\lambda m^{\mathtt{N}}\ c(g_1(m))\dots(g_n(m)) \downarrow [(\mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \dots \sqcup \mathcal{L}_m)_h(z), h((\mathcal{L}_0 \sqcup \mathcal{L}_1 \sqcup \dots \sqcup \mathcal{L}_m)_h(z))]$$

which is the thesis.
ii) $m=0$. Straightforward simplification of the argument for i).

(3) $c : A$, A atomic. By definition 5.4.5

$$[\![c]\!]_s = \langle \lambda h^{\mathtt{N}\to\mathtt{N}} \lambda m^{\mathtt{N}} m, \lambda n^{\mathtt{N}} c\rangle$$

We have therefore to prove that $\lambda h^{\mathtt{N}\to\mathtt{N}}\lambda m^{\mathtt{N}} m$ is a modulus of convergence for $\lambda n^{\mathtt{N}} c$, which is trivially true, and that $\lambda n^{\mathtt{N}} c[s_n] = \lambda n^{\mathtt{N}} c$, which is also trivial. We conclude $[\![c]\!]_s\ \mathsf{gmc}_s\ c$.

(4) $c = \mathsf{R}_T$, R_T recursor constant with $T = A \to (\mathtt{N}\to A\to A)\to \mathtt{N}\to A$. Suppose $\mathcal{I}\ \mathsf{gmc}_s\ u : A$, $\mathcal{L}\ \mathsf{gmc}_s\ v : \mathtt{N}\to A\to A$ and $\langle \mathcal{M}, g\rangle\ \mathsf{gmc}_s\ t : \mathtt{N}$. We have to prove that

$$[\![\mathsf{R}_T]\!]_s \mathcal{I}\mathcal{L}\langle \mathcal{M}, g\rangle = \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})\ \mathsf{gmc}_s\ \mathsf{R}_T uvt$$

where

$$\mathcal{N} := \lambda n^{\mathtt{N}}\ \mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n}) n$$

If we show that for all numerals n, $\mathcal{N}_n\ \mathsf{gmc}_s\ \mathsf{R}_T uvn$, by lemma 5.4.2 we obtain that

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N})\ \mathsf{gmc}_s\ \mathsf{R}_T uvt$$

We prove that by induction on n.
If $n=0$, then

$$\mathcal{N}_0 = \mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n})0 = \mathcal{I}\ \mathsf{gmc}_s\ u = \mathsf{R}_T uv0$$

If $n = \mathsf{S}(m)$, then

$$\begin{aligned}\mathcal{N}_{\mathsf{S}(m)} &= \mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n})\mathsf{S}(m)\\ &= (\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n}) m (\mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n}) m)\\ &= \mathcal{L}\mathcal{M}_{\mathsf{id},m}(\mathsf{R}_U \mathcal{I}(\lambda n^{\mathtt{N}} \mathcal{L}\mathcal{M}_{\mathsf{id},n}) m)\\ &= \mathcal{L}\mathcal{M}_{\mathsf{id},m}\mathcal{N}_m\end{aligned}$$

By induction hypothesis, $\mathcal{N}_m \ \mathsf{gmc}_s \ \mathsf{R}_T uvm$. Moreover, $\mathcal{M}_{\mathsf{id},m} \ \mathsf{gmc}_s \ m$ and by hypothesis $\mathcal{L} \ \mathsf{gmc}_s \ v$. Hence

$$\mathcal{L}\mathcal{M}_{\mathsf{id},m}\mathcal{N}_m \ \mathsf{gmc}_s \ vm(\mathsf{R}_T uvm) = \mathcal{R}_T uv\mathsf{S}(m)$$

which is the thesis.

(5) $c = \mathsf{if}_T$, with $T : \mathtt{Bool} \to A \to A \to A$. Suppose $\mathcal{L}_1 \ \mathsf{gmc}_s \ u_1 : A$, $\mathcal{L}_2 \ \mathsf{gmc}_s \ u_2 : A$ and $\langle \mathcal{M}, g\rangle \ \mathsf{gmc}_s \ t : \mathtt{Bool}$. We have to prove that

$$[\![\mathsf{if}_T]\!]_s\langle \mathcal{M}, g\rangle \mathcal{L}_1\mathcal{L}_2 = \mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) \ \mathsf{gmc}_s \ \mathsf{if}_T t u_1 u_2$$

where

$$\mathcal{N} := \lambda b^{\mathtt{Bool}} \mathsf{if} \ b \ \mathsf{then} \ \mathcal{L}_1 \ \mathsf{else} \ \mathcal{L}_2$$

If we show that for all $a \in \{\mathtt{True}, \mathtt{False}\}$, $\mathcal{N}_a \ \mathsf{gmc}_s \ (\lambda b^{\mathtt{Bool}} \mathsf{if}_T b u_1 u_2)a$, by lemma 5.4.2 we obtain that

$$\mathcal{H}(\langle \mathcal{M}, g\rangle, \mathcal{N}) \ \mathsf{gmc}_s \ (\lambda b^A \mathsf{if}_T b u_1 u_2)t = \mathsf{if}_T t u_1 u_2$$

We prove that by cases.
If $a = \mathtt{True}$, then

$$\mathcal{N}_a = \mathcal{L}_1 \ \mathsf{gmc}_s \ u_1 = (\lambda b^A \mathsf{if}_T b u_1 u_2)a$$

If $a = \mathtt{False}$, then

$$\mathcal{N}_a = \mathcal{L}_2 \ \mathsf{gmc}_s \ u_2 = (\lambda b^A \mathsf{if}_T b u_1 u_2)a$$

Hence, we have the thesis.

□

We are finally ready to define the interpretation of every term of $\mathcal{T}$ in our model of hypernaturals with moduli.

Definition 5.4.6 (Generalized Moduli of Convergence for Terms of $\mathcal{T}$). For every term $v : T$ of system $\mathcal{T}$ and $s \in$w.i., we define a term $[\![v]\!]_s : \mathsf{M}_T$ by induction on v and by cases as follows:

(1) $v = c$, with c constant. We define $[\![c]\!]_s$ as in definition 5.4.5.

(2) $v = x^A$, x variable. Then

$$[\![x^A]\!]_s := x^{\mathsf{M}_A}$$

(3) $v = ut$. Then

$$[\![ut]\!]_s := [\![u]\!]_s[\![t]\!]_s$$

(4) $v = \lambda x^A u$. Then

$$[\![\lambda x^A u]\!]_s := \lambda x^{\mathsf{M}_A} \ [\![u]\!]_s$$

(5) $v = \langle u, t\rangle$. Then

$$[\![\langle u, t\rangle]\!]_s := \langle [\![u]\!]_s, [\![t]\!]_s\rangle$$

(6) $v = \pi_i u$. Then

$$[\![\pi_i u]\!]_s := \pi_i [\![u]\!]_s$$

5.5. Adequacy Theorem

We are now able to prove our main theorem. For every closed term u, $[\![u]\!]_s$ is an inhabitant of the model of hypernaturals with moduli of convergence.

THEOREM 5.5.1 (ADEQUACY THEOREM). *Let $w : A$ be a term of $\mathcal{T}$ and let $x_1^{A_1}, \dots, x_n^{A_n}$ contain all the free variables of w. Then, for all $s \in w.i.$*

$$\lambda x_1^{\mathsf{M}_{A_1}} \dots \lambda x_n^{\mathsf{M}_{A_n}} [\![w]\!]_s \ \mathsf{gmc}_s \ \lambda x_1^{A_1} \dots \lambda x_n^{A_n} w$$

PROOF. Let $t_1 : A_1, \dots, t_n : A_n$ be arbitrary terms. We have to prove that

$$\mathcal{M}_1 \ \mathsf{gmc}_s \ t_1, \dots, \mathcal{M}_n \ \mathsf{gmc}_s \ t_n \implies [\![w]\!]_s[\mathcal{M}_1/x_1^{\mathsf{M}_{A_1}} \dots \mathcal{M}_n/x_n^{\mathsf{M}_{A_n}}] \ \mathsf{gmc}_s \ w[t_1/x_1^{A_1} \dots t_n/x_n^{A_n}]$$

For any term v, we set

$$\overline{v} := v[t_1/x_1^{A_1} \cdots t_n/x_n^{A_n}]$$

and

$$\overline{[\![v]\!]_s} := [\![v]\!]_s[\mathcal{M}_1/x_1^{\mathsf{M}_{A_1}} \dots \mathcal{M}_n/x_n^{\mathsf{M}_{A_n}}]$$

With that notation, we have to prove that $\overline{[\![w]\!]_s} \ \mathsf{gmc}_s \ \overline{w}$. The proof is by induction on w and proceeds by cases, accordingly to the form of w.

(1) $w = c$, with c constant. Since $[\![c]\!]_s$ is closed and c does not have free variables, by proposition 5.4.1

$$\overline{[\![w]\!]_s} = \overline{[\![c]\!]_s} = [\![c]\!]_s \ \mathsf{gmc}_s \ c = \overline{c} = \overline{w}$$

which is the thesis.

(2) $w = x_i^{A_i}$, for some $1 \leq i \leq n$. Then

$$\overline{[\![w]\!]_s} = x_i^{\mathsf{M}_{A_i}}[\mathcal{M}_1/x_1^{\mathsf{M}_{A_1}} \dots \mathcal{M}_n/x_n^{\mathsf{M}_{A_n}}] = \mathcal{M}_i \ \mathsf{gmc}_s \ t_i = x_i^{A_i}[t_1/x_1^{A_1} \cdots t_n/x_n^{A_n}] = \overline{w}$$

which is the thesis.

(3) $w = ut$. By induction hypothesis, $\overline{[\![u]\!]_s} \ \mathsf{gmc}_s \ \overline{u}$ and $\overline{[\![t]\!]_s} \ \mathsf{gmc}_s \ \overline{t}$. So

$$\overline{[\![ut]\!]_s} = \overline{[\![u]\!]_s[\![t]\!]_s} \ \mathsf{gmc}_s \ \overline{u}\overline{t} = \overline{w}$$

which is the thesis.

(4) $w = \lambda x^A u$. Let $t : A$ and suppose $\mathcal{M} \ \mathsf{gmc}_s \ t$. We have to prove that $\overline{[\![w]\!]_s}\mathcal{M} \ \mathsf{gmc}_s \ \overline{w}t$. By induction hypothesis

$$\overline{[\![\lambda x^A u]\!]_s}\mathcal{M} = (\lambda x^{\mathsf{M}_A} \overline{[\![u]\!]_s})\mathcal{M} = \overline{[\![u]\!]_s}[\mathcal{M}/x^{\mathsf{M}_A}] \ \mathsf{gmc}_s \ \overline{u}[t/x^A] = \overline{w}t$$

which is the thesis.

(5) $w = \langle u_0, u_1 \rangle$. By induction hypothesis, $\overline{[\![u_0]\!]_s} \ \mathsf{gmc}_s \ \overline{u}_0$ and $\overline{[\![u_1]\!]_s} \ \mathsf{gmc}_s \ \overline{u}_1$. Therefore, for $i = 0, 1$

$$\pi_i \overline{[\![\langle u_0, u_1 \rangle]\!]_s} = \pi_i \langle \overline{[\![u_0]\!]_s [\![u_1]\!]_s} \rangle = \overline{[\![u_i]\!]_s} \ \mathsf{gmc}_s \ \overline{u}_i = \pi_i \overline{w}$$

which is the thesis.

(6) $w = \pi_i u$, with $i \in \{0,1\}$. By induction hypothesis, $\overline{[\![u]\!]_s}$ gmc_s $\overline{u}$. Therefore,

$$\overline{[\![\pi_i u]\!]_s} = \pi_i \overline{[\![u]\!]_s} \ \mathsf{gmc}_s \ \pi_i \overline{u} = \overline{w}$$

which is the thesis.

□

5.6. Consequences of the Adequacy Theorem

In this section, we spell out most interesting consequences of adequacy theorem.

5.6.1. Weak Convergence Theorem. We can finally prove the constructive version of theorem 5.1.1, our main goal. The following theorem is even stronger of the previously enunciated theorem 5.3.1, because it states that one can find moduli of convergence for any uniformly defined collection of terms.

THEOREM 5.6.1 (WEAK CONVERGENCE THEOREM FOR COLLECTION OF TERMS). *Let $t : \mathtt{N} \to (\mathtt{N} \to \mathtt{N}) \to \mathtt{S}$ be a closed term of $\mathcal{T}$ not containing Φ, with $\mathtt{S}$ atomic type. Then we can effectively define a closed term $\mathcal{M} : \mathtt{N} \to (\mathtt{N} \to (\mathtt{N} \to \mathtt{N})) \to (\mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})$ of $\mathcal{T}$, such that for all $s : \mathtt{N} \to (\mathtt{N} \to \mathtt{N})$, $s \in w.i.$ and numerals n, $\mathcal{M}_n s$ is a modulus of convergence for $\lambda m^{\mathtt{N}} t_n(s_m)$.*

PROOF. Let

$$\mathcal{M} := \lambda y^{\mathtt{N}} \lambda s^{\mathtt{N}\to(\mathtt{N}\to\mathtt{N})} \pi_0((\lambda x^{\mathsf{M}_{\mathtt{N}}} [\![t_{x^{\mathtt{N}}} \Phi]\!]_s) \mathcal{M}_{\mathsf{id},y})$$

By the adequacy theorem 5.5.1,

$$\lambda x^{\mathsf{M}_{\mathtt{N}}} [\![t_{x^{\mathtt{N}}} \Phi]\!]_s \ \mathsf{gmc}_s \ \lambda x^{\mathtt{N}} t_{x^{\mathtt{N}}} \Phi$$

Since for every numeral n, $\mathcal{M}_{\mathsf{id},n}$ gmc_s n, we have

$$(\lambda x^{\mathsf{M}_{\mathtt{N}}} [\![t_{x^{\mathtt{N}}} \Phi]\!]_s) \mathcal{M}_{\mathsf{id},n} \ \mathsf{gmc}_s \ t_n \Phi$$

By definition of generalized modulus of convergence and of $\mathcal{M}$, $\mathcal{M}_n s$ is a modulus of convergence for $\lambda m^{\mathtt{N}} t_n \Phi[s_m] = \lambda m^{\mathtt{N}} t_n(s_m)$.

□

5.6.2. Learning Based Realizability and Provably Total Functions of PA. If σ is a state of knowledge, we mantain the notation

$$u[\sigma] := [\varphi_P \sigma / \Phi_P \ \chi_P \sigma / \mathsf{X}_P \ \mathsf{add}_P \sigma / \mathsf{Add}_P]$$

of chapter 3, since there is no confusion with the notation $u[v]$ of the previous section, which was defined for v of type $\mathtt{N} \to \mathtt{N}$.

THEOREM 5.6.2 (ZERO THEOREM FOR COLLECTIONS OF TERMS OF $\mathcal{T}_{\text{CLASS}}$). *Let $t : \mathtt{N} \to \mathtt{S}$ be a closed term of $\mathcal{T}_{Class}$, where $\mathtt{S}$ is the atomic type of knowledge states. Then, there exists a term $\mathsf{Zero} : \mathtt{N} \to \mathtt{S}$ of $\mathcal{T}_{Learn}$ such that for every numeral n, $t_n[\mathsf{Zero}_n] = \varnothing$.*

PROOF. We may assume t contains as oracles only constants $\mathsf{X}_P, \Phi_P, \mathsf{Add}_P$ for some fixed predicate P. The general case is analogous and involves only a little bit more of trivial coding.
In the first place, we have to carry out some simple coding. We have to represent states

of knowledge by terms of type $\mathtt{N} \to \mathtt{N}$. This is straightforward since a state represents a function over $\mathbb{N}$. Define

$$f := \lambda\sigma^{\mathtt{S}}\lambda n^{\mathtt{N}}\mathsf{if}\ \chi_P\sigma n\ \mathsf{then}\ \varphi_P\sigma n + 1\ \mathsf{else}\ 0$$

f takes a state σ and returns the function coding it; if n is a numeral, when $\chi_P\sigma n = \mathtt{True}$, $f_\sigma n$ returns $\varphi_P\sigma n + 1$ and not just $\varphi_P\sigma n$, in order to ensure that 0 is returned only when it is just a trivial value, i.e. when $\chi_P\sigma = \mathtt{False}$.
Given a term $g : \mathtt{N} \to \mathtt{N}$, intended to represent a state σ, define terms $\varphi_P^g, \chi_P^g, \mathsf{add}_P^g$, intended to code respectively $\varphi_P\sigma, \chi_P\sigma, \mathsf{add}_P\sigma$, as follows

$$\varphi_P^g := \lambda n^{\mathtt{N}}\mathsf{if}\ g(n) = 0\ \mathsf{then}\ 0\ \mathsf{else}\ g(n) - 1$$
$$\chi_P^g := \lambda n^{\mathtt{N}}\mathsf{if}\ g(n) = 0\ \mathsf{then}\ \mathtt{False}\ \mathsf{else}\ \mathtt{True}$$
$$\mathsf{add}_P^g := \lambda n^{\mathtt{N}}\lambda m^{\mathtt{N}}\mathsf{if}\ g(n) = 0\ \mathsf{then}\ \mathsf{add}_P\varnothing nm\ \mathsf{else}\ \varnothing$$

Moreover, for every term u, define

$$u^g := u[\varphi_P^g/\Phi_P\ \chi_P^g/\mathsf{X}_P\ \mathsf{add}_P^g/\mathsf{Add}_P]$$

It is easy to see that for all terms u and all states σ, $u[\sigma] = u^{f_\sigma}$.
We can now give the important part of the argument. Fix a numeral n. Let $\sigma_m := \varnothing$ and $\sigma_{m+1} := \sigma_m \Cup t_n[\sigma_m]$ be a recursive definition of a sequence of states (which can be coded in $\mathcal{T}$). Our goal is to write down a term of system $\mathcal{T}_{\text{Learn}}$ which is able to find a state σ_{k+1} such that $t_n[\sigma_k] = t_n[\sigma_{k+1}]$: as we will see, this condition implies $t[\sigma_{k+1}] = \varnothing$. Let

$$\mathcal{U} := \lambda m^{\mathtt{N}}\lambda g^{\mathtt{N}\to\mathtt{N}} t_m^g$$

By the weak convergence theorem 5.6.1 applied to $\mathcal{U}$, there exists a term $\mathcal{M}$ of $\mathcal{T}_{\text{Learn}}$ such that, for every numeral l and $s \in$ w.i., $\mathcal{M}_l s$ is a modulus of convergence for $\lambda m^{\mathtt{N}}\mathcal{U}_l(s_m)$. Set $s := (\lambda m^{\mathtt{N}} f_{\sigma_m})$. Then $s \in$ w.i., since $\sigma_0 \leq \sigma_1 \leq \sigma_2 \cdots$. Therefore $\mathcal{M}_n s$ is a modulus of convergence for $\lambda m^{\mathtt{N}}\,\mathcal{U}_n(s_m)$. If we choose $h := \lambda m^{\mathtt{N}} m+1$ and set $k := \mathcal{M}_n s h$, we have that $\mathcal{U}_n(s_k) = \mathcal{U}_n(s_{k+1})$ by definition of modulus of convergence. We thus obtain, by definition of $\mathcal{U}$ ad s, that $t_n^{f_{\sigma_k}} = t_n^{f_{\sigma_{k+1}}}$ and so $t_n[\sigma_k] = t_n[\sigma_{k+1}]$. Let

$$\mathsf{Zero} := \lambda m^{\mathtt{N}}\sigma_{(\mathcal{M}_m s h)+1}$$

We have

$$\begin{aligned}
\mathsf{Zero}_n \Cup t_n[\mathsf{Zero}_n] &= \sigma_{k+1} \Cup t_n[\sigma_{k+1}] \\
&= (\sigma_k \Cup t_n[\sigma_k]) \Cup t_n[\sigma_{k+1}] \\
&= (\sigma_k \Cup t_n[\sigma_k]) \Cup t_n[\sigma_k] \\
&= \sigma_k \Cup t_n[\sigma_k] \\
&= \sigma_{k+1} \\
&= \mathsf{Zero}_n
\end{aligned}$$

Since $t_n[\mathsf{Zero}_n]$ is consistent and disjoint with Zero_n, we conclude $t_n[\mathsf{Zero}_n] = \varnothing$ and obtain the thesis.

□

As anticipated in the introduction, from learning based realizers we can extract algorithms of system T.

THEOREM 5.6.1 (PROGRAM EXTRACTION VIA LEARNING BASED REALIZABILITY). *Let t be a term of $\mathcal{T}_{Class}$ and suppose that $t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} P x y$, with P atomic. Then, from t one can effectively define a term u of Gödel's system T such that for every numeral n, $Pn(un) = \mathtt{True}$.*

PROOF. Let

$$v := \lambda m^{\mathtt{N}} \pi_1(tm)$$

v is of type $\mathtt{N} \to \mathtt{S}$. By theorem 5.6.2, there exists a term $\mathsf{Zero} : \mathtt{N} \to \mathtt{S}$ of $\mathcal{T}_{\mathrm{Learn}}$ such that $v_n[\mathsf{Zero}_n] = \varnothing$ for every numeral n. Define

$$w := \lambda m^{\mathtt{N}} (\pi_0(tm)[\mathsf{Zero}_m])$$

and fix a numeral n. By unfolding the definition of realizability with respect to the state Zero_n, we have that

$$tn \Vdash_{\mathsf{Zero}_n} \exists y^{\mathtt{N}} P n y$$

and hence

$$\pi_1(tn) \Vdash_{\mathsf{Zero}_n} \; Pn(wn)$$

that is to say

$$v_n[\mathsf{Zero}_n] = \varnothing \implies Pn(wn) = \mathtt{True}$$

and therefore

$$Pn(wn) = \mathtt{True}$$

We observe that $w : \mathtt{N} \to \mathtt{N}$ and w is a term of $\mathcal{T}_{\mathrm{Learn}}$. By standard coding of states into natural numbers and of all other constants of $\mathcal{T}_{\mathrm{Learn}}$ into terms of Gödel's T, one can code every term of $\mathcal{T}_{\mathrm{Learn}}$ into Gödel's T. Hence, there exists a term u of Gödel's T such that for all numerals n, $u(n) = w(n)$, which is the thesis.

□

REMARK 5.6.3 . We point out that the term u of theorem 5.6.1, modulo some trivial coding of states into numbers, bears a strong resemblance to the term t from which it is defined. In particular, u is straightforwardly obtained from a modulus of convergence $\mathcal{M}$ that carries the constructive information associated to the convergence of t. In turn, $\mathcal{M}$ is obtained via the translation $[\![\_]\!]_s$, which just replaces type-A constants and variables of t with new terms of type M_A. As an instance, recursion constants R_A are replaced with recursion constants $\mathsf{R}_{\mathsf{M}_A}$. Therefore, the type of recursion goes through a constant increase of 2, since M_A is obtained by changing the basic types C with M_C.
We conjecture it is not possible to amend our translation as to preserve the types of recursion constants. This should be due to Avigad's theorem: if one is able to find zeros of finite update procedures, than one can compute every provably total function of PA. But in the next subsections, we show how to compute finite zeros of update procedures in Gödel's system T, thanks to moduli of convergence. If the translation $[\![\_]\!]_s$ did not increase the recursion type, then the term computing the zero of an update procedure would have the same recursion level of the latter. But since primitive recursive functionals are enough as update procedures, one would get a contradiction to Avigad's theorem, because one could interpret all provably total functions of Arithmetic with primitive recursion.

We are now able to prove a version of the classic theorem of Gödel, characterizing the class of functions provably total in PA as the class of functions representable in system T.

THEOREM 5.6.4 (PROVABLY TOTAL FUNCTIONS OF PA). *If* $\mathsf{PA} \vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$, *then there exists a term* u *of Gödel's system* T *such that for every numeral* n, $Pn(un) = \mathtt{True}$.

PROOF. Starting from the assumption that

$$\mathsf{PA} \vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$$

by Kolmogorov double negation translation (see for instance [43]), we have that

$$\mathsf{HA} \vdash \forall x^{\mathtt{N}} \neg\neg \exists y^{\mathtt{N}} Pxy$$

Therefore

$$\mathsf{HA} + \mathsf{EM}_1 \vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$$

and so there is a term t of $\mathcal{T}_{\text{Class}}$ such that

$$t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$$

By theorem 5.6.1, there exists a term u of Gödel's system T such that for all numerals n

$$Pnu(n) = \mathtt{True}$$

□

5.6.3. Zeros for Update Procedures. Thanks to the adeguacy theorem, we are able to give a new constructive proof of Avigad's theorem for update procedures. Here, we give a slightly different definition of update procedure. This is not a limitation, since the update procedures which are *actually* used by Avigad [5] in proving 1-consistency of PA still fall under the following definition.

DEFINITION 5.6.1 (UPDATE OPERATOR, TYPED UPDATE PROCEDURE). Fix a primitive recursive bijective coding $|\_| : (\mathbb{N}^3 \cup \{\varnothing\}) \to \mathbb{N}$ of $\varnothing$ and of triples of natural numbers into natural numbers. Define a binary operation $\oplus$ which combine functions $f : \mathtt{N} \to \mathtt{N}$ and codes of the form $|(1, n, m)|$ of pairs of natural numbers and returns a function $\mathtt{N} \to \mathtt{N}$ as follows

$$f \oplus |(1, m, n)| := \lambda x^{\mathtt{N}} \mathsf{if}\ x = m\ \mathsf{then}\ n\ \mathsf{else}\ f(x)$$

For convenience, define also $f \oplus |\varnothing| = f$.

A *typed update procedure of ordinal* $k \in \mathbb{N}$ (also said a k-ary typed update procedure) is a term $\mathcal{U} : (\mathtt{N} \to \mathtt{N})^k \to \mathtt{N}$ of Gödel's T such that the following holds:

(1) for all sequences $f = f_1, \ldots, f_k$ of closed type-$\mathtt{N} \to \mathtt{N}$ terms of T, $\mathcal{U}f = |(i, n, m)| \implies 1 \leq i \leq k$.

(2) for all sequences $f = f_1, \ldots, f_k$ and $g = g_1, \ldots, g_k$ of closed type-$\mathtt{N} \to \mathtt{N}$ terms of T and for all $1 \leq i < k$, if

   i) for all $j < i$, $f_j = g_j$;

   ii) $\mathcal{U}f = |(i, n, m)|$, $g_i(n) = m$ and $\mathcal{U}g = |(i, h, l)|$

   then $h \neq n$.

If $\mathcal{U}$ is a k-ary typed update procedure, a *zero* for $\mathcal{U}$ is a sequence $f = f_1, \ldots, f_k$ of closed type-$\mathtt{N} \to \mathtt{N}$ terms of T such that $\mathcal{U}f = |\varnothing|$.

Every unary update procedure gives rise to a learning process, i.e. a weakly increasing chain of functions.

PROPOSITION 5.6.5 (LEARNING PROCESSES FROM UNARY UPDATE PROCEDURES). *Let $\mathcal{U}$ be a unary update procedure and define by recursion $s_0 := 0^{\mathtt{N}\to\mathtt{N}} := \lambda x^{\mathtt{N}} 0$ and $s_{k+1} := s_k \oplus \mathcal{U}s_k$. Then $s \in$w.i..*

PROOF. Suppose $s_i(n) = m \neq 0$. We have to prove that for all j, $s_{i+j}(n) = m$. We proceed by induction on j. Suppose $j > 0$. Since $s_0 = 0^{\mathtt{N}\to\mathtt{N}}$ and $s_i(n) \neq 0$, it must be that for some $i_0 < i$, $\mathcal{U}s_{i_0} = |(1, n, m)|$. By induction hypothesis, $s_{i+j-1}(n) = m$. By definition 5.6.1, point 2-ii), $\mathcal{U}s_{i+j-1} \neq |(1, n, l)|$ for all l. Since

$$s_{i+j} = s_{i+j-1} \oplus \mathcal{U}s_{i+j-1}$$

it must be that $s_{i+j}(n) = m$.

□

We first prove that unary typed update procedures have zeros.

THEOREM 5.6.6 (ZERO THEOREM FOR UNARY TYPED UPDATE PROCEDURES). *Let $\mathcal{U} : (\mathtt{N} \to \mathtt{N})^{k+1} \to \mathtt{N}$ be a term of T such that for all closed type-$\mathtt{N} \to \mathtt{N}$ terms $f_1, \ldots, f_k$ of T, $\mathcal{U}f_1 \ldots f_k$ is a typed unary update procedure. Then one can constructively define a closed term $\varepsilon : (\mathtt{N} \to \mathtt{N})^k \to (\mathtt{N} \to \mathtt{N})$ of T such that for all closed type-$\mathtt{N} \to \mathtt{N}$ terms $f_1, \ldots, f_k$ of T*

$$\mathcal{U}f_1 \ldots f_k(\varepsilon f_1 \ldots f_k) = |\varnothing|$$

PROOF. First, for any term $h : \mathtt{N} \to \mathtt{N}$, define

$$\mathcal{L}^h := \lambda\langle \mathcal{M}, g\rangle^{\mathsf{M}_{\mathtt{N}}} \langle \mathcal{M}, \lambda n^{\mathtt{N}} h(g(n))\rangle$$

The same proof of proposition 5.4.1 (in the case of constants of type $\mathtt{N} \to \mathtt{N}$) shows that for all closed terms h of T and $s \in$w.i., $\mathcal{L}^h$ gmc_s h. Define

$$\mathcal{N}_s := \lambda h_1^{\mathtt{N}\to\mathtt{N}} \ldots \lambda h_k^{\mathtt{N}\to\mathtt{N}} [\![\mathcal{U}]\!]_s \mathcal{L}^{h_1} \ldots \mathcal{L}^{h_k} [\![\Phi]\!]_s$$

and fix closed type-$\mathtt{N} \to \mathtt{N}$ terms $f_1, \ldots, f_k$ of T. By the adeguacy theorem 5.5.1, for all $s \in$w.i., $[\![\mathcal{U}]\!]_s$ gmc_s $\mathcal{U}$ and hence

$$\mathcal{M}_s := \mathcal{N}_s f_1 \ldots f_k \ \mathsf{gmc}_s \ \mathcal{U}f_1 \ldots f_k \Phi$$

So, for all $s \in$ w.i., $\pi_0(\mathcal{M}_s)$ is a modulus of convergence for $\lambda m^{\mathtt{N}} \mathcal{U}f_1 \ldots f_k(s_m)$. Define by recursion a term s such that $s_0 := 0^{\mathtt{N}\to\mathtt{N}}$ and $s_{n+1} := s_n \oplus \mathcal{U}f_1 \ldots f_k s_n$. Then $s \in$w.i. by proposition 5.6.5, since $\mathcal{U}f_1 \ldots f_k$ is a typed unary update procedure. So $\pi_0(\mathcal{M}_s)$ is a modulus of convergence for $\lambda m^{\mathtt{N}}$ $\mathcal{U}f_1 \ldots f_k(s_m)$. If we choose $h := \lambda m^{\mathtt{N}} m + 1$ and set $j := \pi_0(\mathcal{M}_s)h$, we have that

$$\mathcal{U}f_1 \ldots f_k(s_j) = \mathcal{U}f_1 \ldots f_k(s_{j+1})$$

by definition of modulus of convergence. So let

$$\varepsilon f_1 \dots f_k := s_{(\pi_0(\mathcal{M}_s)h)+1}$$

Then

$$\begin{aligned} s_{j+2} &= s_{j+1} \oplus \mathcal{U} f_1 \dots f_k(s_{j+1}) \\ &= (s_j \oplus \mathcal{U} f_1 \dots f_k(s_j)) \oplus \mathcal{U} f_1 \dots f_k(s_{j+1}) \\ &= (s_j \oplus \mathcal{U} f_1 \dots f_k(s_j)) \oplus \mathcal{U} f_1 \dots f_k(s_j) \\ &= s_j \oplus \mathcal{U} f_1 \dots f_k(s_j) \\ &= s_{j+1} \end{aligned}$$

and hence it must be that

$$\mathcal{U} f_1 \dots f_k(\varepsilon f_1 \dots f_k) = \mathcal{U} f_1 \dots f_k(s_{j+1}) = |\varnothing|$$

which is the thesis.

□

We are now able to prove the zero theorem for n-ary typed update procedures, following the idea of Avigad's original construction.

THEOREM 5.6.7 (ZERO THEOREM FOR N-ARY TYPED UPDATE PROCEDURES). *Let* $\mathcal{U} : (\mathtt{N} \to \mathtt{N})^k \to \mathtt{N}$ *be a typed update procedure of ordinal* $k \in \mathbb{N}$. *Then one can constructively define terms* $\varepsilon_1, \dots, \varepsilon_k$ *of Gödel's* T *such that*

$$\mathcal{U} \varepsilon_1 \dots \varepsilon_k = |\varnothing|$$

PROOF. By induction on k. The case $k = 1$ has been treated in theorem 5.6.6. Therefore, suppose $k > 1$. Define

$$\mathcal{U}_k := \lambda g_1^{\mathtt{N}\to\mathtt{N}} \dots \lambda g_{k-1}^{\mathtt{N}\to\mathtt{N}} \mathsf{if}\ \mathcal{U} g_1 \dots g_{k-1} = |(k,n,m)|\ \mathsf{then}\ |(1,n,m)|\ \mathsf{else}\ |\varnothing|$$

Since for all closed type-$\mathtt{N} \to \mathtt{N}$ terms $f_1, \dots, f_{k-1}$ of T, $\mathcal{U}_k f_1 \dots f_{k-1}$ is a unary typed update procedure, by theorem 5.6.6, we can constructively define a term ε_k of T such that for all closed type-$\mathtt{N} \to \mathtt{N}$ terms $f_1, \dots, f_{k-1}$ of T

$$\mathcal{U}_k f_1 \dots f_{k-1}(\varepsilon_k f_1 \dots f_{k-1}) = |\varnothing|$$

and hence

$$\mathcal{U} f_1 \dots f_{k-1}(\varepsilon_k f_1 \dots f_{k-1}) \neq |(k,n,m)|$$

This implies that

$$\lambda g_1^{\mathtt{N}\to\mathtt{N}} \dots \lambda g_{k-1}^{\mathtt{N}\to\mathtt{N}} \mathcal{U} g_1 \dots g_{k-1}(\varepsilon g_1 \dots g_{k-1})$$

is a typed update procedure of ordinal $k-1$. By induction hypothesis, we can constructively define terms $\varepsilon_1, \dots, \varepsilon_{k-1}$ of Gödel's T such that

$$\mathcal{U} \varepsilon_1 \dots \varepsilon_{k-1}(\varepsilon_k \varepsilon_1 \dots \varepsilon_{k-1}) = |\varnothing|$$

which is the thesis.

□

An important corollary of theorem 5.6.7, is the termination of the epsilon substitution method for first order Peano Arithmetic.

THEOREM 5.6.8 (TERMINATION OF EPSILON SUBSTITUTION METHOD FOR PA). *The H-process (as defined in Mints [35]) of the epsilon substitution method for* PA *always terminates.*

PROOF. See Avigad [5] □

CHAPTER 6

Learning in Predicative Analysis

Abstract. We give an axiomatization of the concept of learning, as it implicitly appears in various computational interpretations of Predicative classical second order Arithmetic. We achieve our result by extending Avigad's notion of update procedure to the transfinite case.

6.1. Introduction

The aim of this chapter 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 learning based realizability to predicative fragments of Analysis: a possible path to follow is the one suggested here. In particular, we describe the learning processes that arise when extending the approach of learning based realizability to predicative Arithmetic and prove their termination. This is achieved by introducing the notion of transfinite update procedure.

Secondly, we continue the work of Avigad on update procedures [5] and - as anticipated - extend them to the transfinite case. The concept of transfinite update procedure may be seen as an axiomatization of learning as implicitly used in the epsilon substitution method formulated in the work of Mints et al ([35], [36]). The notion is useful to understand the core of the epsilon method and its fundamental ideas without having to deal with the complicated formalism and non relevant details. Moreover, as interesting byproduct of the conceptual analysis of the epsilon method through transfinite update procedures, one can provide a combinatorial statement that is equivalent to the 1-consistency of various fragments of predicative Arithmetic in a very scalable way. In particular, the informal statement "$\mathsf{U}(\alpha)$: every transfinite update procedure of ordinal less than α has a finite zero" very rapidly acquires logical complexity even at small ordinals. For example:

(1) $\mathsf{U}(2)$ corresponds to the 1-consistency of $\mathsf{HA} + \mathsf{EM}_1$.
(2) $\mathsf{U}(n+1)$, with $n \in \omega$, corresponds to the 1-consistency of $\mathsf{HA} + \mathsf{EM}_n$ (excluded middle over Σ^0_n formulas).
(3) $\mathsf{U}(\omega)$ corresponds to the 1-consistency of PA
(4) $\mathsf{U}(\omega \cdot 2)$ corresponds to the 1-consistency of EA (Elementary Analysis)
(5) $\mathsf{U}(\omega^\omega)$ corresponds to the 1-consistency of PA^2 plus Δ^1_1-comprehension axiom.

(1), (4) are treated in this thesis, (3) has been proved by Avigad [5], (2) will follow by extending learning based realizability. (5) should follow by straightforward extension of the methods we will use to prove (4). In order to make precise the statement $\mathsf{U}(\alpha)$ and prove refined versions of (1)-(5), one has to choose a formal system in to which represent update procedures. All the update procedures used in this thesis may be assumed to be represented

in system T.

Plan of the Chapter. In section §6.2 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 §6.3 we explain 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 more constructive proof of the existence of finite zeros and the learning processes are "optimal", in the sense that one could provide constructively the expected ordinal bounds to their length and of the size of finite zeros (by applying techniques of Mints [35])

In sections §6.4 and §6.5 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.

In section §6.6 we devote ourselves to a case study: we show that $\mathsf{U}(\omega \cdot 2)$ implies the 1-consistency of EA by proving that it implies the termination of H-processes of the epsilon substitution method for EA.

6.2. Transfinite Update Procedures for Predicative Analysis

From the computational point of view, classical predicative second order Arithmetic poses very difficult problems. Axioms of comprehension ask for functions g able to decide truth of formulas:

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

Axioms of (countable) choice ask for functions g computing existential witnesses of truth of formulas:

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

A Kleene-style realizability interpretation for even the most simple form of the excluded middle

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

asks for deciding the truth of semidecidable formulas of the form $\exists x^{\mathtt{N}}\ Pnx$, with P decidable.

In general, learning based computational interpretations of predicative fragments of classical analysis (such as our learning based realizability, epsilon substitution method, update procedures, Herbrand analysis) provide answers to the above computational challenges by the following three-stage pattern:

(1) They identify a sequence F - possibly transfinite - of non computable functions $\mathbb{N} \to \mathbb{N}$.

(2) They define classical witnesses for provable formulas by using programs recursive in F.

(3) They define learning procedures through which it is possible to find, for every particular computation, a suitable finite approximation of the functions of F such that one can effectively compute the witnesses defined at stage two.

The functions in the F 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

F as a sequence of infinite stacks of numbers, the learning process of point (3) finds a "vertical" approximation of F: functions of F are infinite stacks of numbers whereas their finite approximations are finite stacks. Moreover, a crucial point is that the sequence F is not an arbitrary sequence. In a sense, F is also "horizontally" approximated: for every ordinal α, the recursion theoretic Turing degree of F_α is approximated by the degrees of F_β, for $\beta < \alpha$. This property is very important: in this way, the values of the functions in F can be gradually approximated and learned.

More precisely, F can be seen a sequence of functions obtained by transfinite iteration of recursion theoretic *jump* operator (see for example Odifreddi [38]). That is, for every β, if β is a successor, F_β has the same Turing degree of an oracle for the halting problem for the class of functions recursive in $F_{\beta-1}$ (jump); if β is limit, F_β has the same Turing degre of the function mapping the code of a pair (α, n), with $\alpha < \beta$, into $F_\alpha(n)$ (join or β-jump). A fundamental property of such a sequence is that the assertion $F_\beta(n) = m$ depends only on the values of the functions F_α, for $\alpha < \beta$, and the values of F_β are learnable in the limit[1] by a program g recursive in the join of F_α for $\alpha < \beta$, which is a guarantee that 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 EA. A complete treatment is postponed to section 6.6.

Example 6.2.1 (Elementary Analysis). Consider a subsystem 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

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

then the truth of

$$\exists x^{\mathbf{N}} \phi(n, x)$$

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 $t(n, m) = \mathtt{True}$. 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

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

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

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

a program t_n such that

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

[1] In the sense of Gold [22]: $F_\beta(n) = m \iff \lim_{k\to\infty} g(n, k) = m$

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 F of non computable functions F of length $\omega + \omega$. All the programs that we have described are recursive in some initial segment of F.

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

Definition 6.2.1 (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

i) for every $\gamma < \beta$, $f_\gamma = g_\gamma$;

ii) $\mathcal{U}f = (\beta, n, m)$, $g_\beta(n) = m$ and $\mathcal{U}g = (\beta, i, j)$;

then $i \neq n$.

The concept of transfinite update procedure is a generalization of Avigad's notion of update procedure [5]. 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 F; 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 F_β and must be updated as to output m on input n. Condition (2) in definition 6.2.1 is a little bit stronger than Avigad's requirement, which would be:

(2)' For all functions $f, g : \alpha \to (\mathbb{N} \to \mathbb{N})$ and every ordinal $\beta \in \alpha$, if

i) for every $\gamma < \beta$, $f_\gamma = g_\gamma$;

ii) $\mathcal{U}f = (\beta, n, m)$, $g_\beta n = m$, $f_\beta \leq g_\beta$ and $\mathcal{U}g = (\beta, i, j)$ (where $f_\beta \leq g_\beta$ is defined as $f_\beta(x) \neq 0$ implies $f_\beta(x) = g_\beta(x)$);

then $i \neq n$.

But in fact, the update procedures which are *actually* used by Avigad [5] in proving 1-consistency of PA still fall under our definition.
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 $F_\beta(n) = m$; so if g_β is a candidate approximation of F_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 sequence F 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 F_β is learned. This is what we call "learning by counterexamples": from every failure a new positive fact is acquired. We have studied an instance of this kind of learning in the chapter on learning based realizability for $\mathsf{HA} + \mathsf{EM}_1$ (for the case of $\alpha = 1$), when we defined realizability for atomic formulas: in that case the pair (n, m) was produced by the realizer of the excluded middle. We will see another instance in section 6.6 in the case $\alpha = \omega + k$, with $k \in \mathbb{N}$: the triple (β, n, m) will be produced through the evaluation of axioms for epsilon terms.

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

Definition 6.2.2 (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 into 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 6.2.3 (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. □

6.3. 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. [35]). The proof of termination is non-constructive and is similar to the one in Mints et al. [35] (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 some layer all the higher ones collapse. We define an update operator $\oplus$ that performs those operations.

Definition 6.3.1 (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 6.3.2 (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:

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

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 indexes.

DEFINITION 6.3.3 . 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 follow

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

(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.

FACT 1 . 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 (for definition of $g_{<\beta}$, recall definition 6.2.2). But the greater the ordinal β the easier 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 [5] to the transfinite case.

We now prove the reduction lemma in the limit case.

LEMMA 6.3.4 (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. (1) Let n be the smallest among the i such that $f = \mathcal{U}_g^{(i)}$. If $k < n$ and $\mathcal{U}_g(\mathcal{U}_g^{(k)}) = \emptyset$, then

$$\mathcal{U}_g^{(k+1)} = \mathcal{U}_g^{(k)} \oplus \mathcal{U}_g(\mathcal{U}_g^{(k)}) = \mathcal{U}_g^{(k)}$$

So we have that for all $k < n$, $\mathcal{U}_g(\mathcal{U}_g^{(k)}) \neq \emptyset$. Since β is a limit ordinal and $\mathcal{U}$ is continuous, there exists a $\gamma < \beta$ such that for every $k \leq n$

$$\mathcal{U}(g * \mathcal{U}_g^{(k)}) = \mathcal{U}(g_{<\gamma} * 0^{\beta-\gamma} * \mathcal{U}_g^{(k)})$$

and

$$\mathcal{U}(g_{<\gamma} * 0^{\beta-\gamma} * f) = \langle \delta, n, m \rangle \wedge \delta < \beta \implies \delta < \gamma \tag{6.1}$$

We prove by induction on $k \leq n$ that

$$\mathcal{U}_{g_{<\gamma}}^{(k)} = 0^{\beta-\gamma} * \mathcal{U}_g^{(k)}$$

which is the thesis. If $k = 0$,

$$\mathcal{U}_{g_{<\gamma}}^{(0)} = 0^{\alpha-\gamma} = 0^{\beta-\gamma} * 0^{\alpha-\beta} = 0^{\beta-\gamma} * \mathcal{U}_g^{(0)}$$

If $k + 1 \leq n$, then for some δ, l, m

$$\mathcal{U}_g(\mathcal{U}_g^{(k)}) = \langle \delta, l, m \rangle$$

Then, by definition of $\mathcal{U}_g$,

$$\mathcal{U}(g * \mathcal{U}_g^{(k)}) = \langle \beta + \delta, l, m \rangle$$

and hence

$$\mathcal{U}(g_{<\gamma} * 0^{\beta-\gamma} * \mathcal{U}_g^{(k)}) = \langle \beta + \delta, l, m \rangle = \langle \gamma + (\beta - \gamma) + \delta, l, m \rangle$$

Therefore, by definition of $\mathcal{U}_{g_{<\gamma}}$

$$\mathcal{U}_{g_{<\gamma}}(0^{\beta-\gamma} * \mathcal{U}_g^{(k)}) = \langle (\beta - \gamma) + \delta, l, m \rangle$$

By the help of induction hypothesis, we conclude that

$$\begin{aligned}
\mathcal{U}_{g_{<\gamma}}^{(k+1)} &= \mathcal{U}_{g_{<\gamma}}^{(k)} \oplus \mathcal{U}_{g_{<\gamma}}(\mathcal{U}_{g_{<\gamma}}^{(k)}) \\
&= \mathcal{U}_{g_{<\gamma}}^{(k)} \oplus \mathcal{U}_{g_{<\gamma}}(0^{\beta-\gamma} * \mathcal{U}_g^{(k)}) \\
&= \mathcal{U}_{g_{<\gamma}}^{(k)} \oplus \langle (\beta - \gamma) + \delta, l, m \rangle \\
&= (0^{\beta-\gamma} * \mathcal{U}_g^{(k)}) \oplus \langle (\beta - \gamma) + \delta, l, m \rangle \\
&= 0^{\beta-\gamma} * (\mathcal{U}_g^{(k)} \oplus \langle \delta, l, m \rangle) \\
&= 0^{\beta-\gamma} * (\mathcal{U}_g^{(k)} \oplus \mathcal{U}_g(\mathcal{U}_g^{(k)})) \\
&= 0^{\beta-\gamma} * \mathcal{U}_g^{(k+1)}
\end{aligned}$$

(2) We continue the previous proof. Suppose that f is also a finite zero of $\mathcal{U}_g$. If

$$\emptyset = \mathcal{U}(g * f) = \mathcal{U}(g_{<\gamma} * 0^{\beta-\gamma} * f)$$

then by definition of $\mathcal{U}_{g_{<\gamma}}$

$$\mathcal{U}_{g_{<\gamma}}(0^{\beta-\gamma} * f) = \emptyset$$

Therefore, suppose

$$\langle \delta, l, m\rangle = \mathcal{U}(g * f) = \mathcal{U}(g_{<\gamma} * 0^{\beta-\gamma} * f)$$

Since $\mathcal{U}_g(f) = \emptyset$, by definition of $\mathcal{U}_g$ we have that $\delta < \beta$. By 6.1, we have also that $\delta < \gamma$. So, by definition of $\mathcal{U}_{g_{<\gamma}}$

$$\mathcal{U}_{g_{<\gamma}}(0^{\beta-\gamma} * f) = \emptyset$$

which is the thesis.

□

We now prove the reduction lemma in the successor case.

LEMMA 6.3.5 (REDUCTION LEMMA, SUCCESSOR CASE). *Let $\mathcal{U}$ be an update procedure of ordinal α. Define $g : \mathbb{N} \to \mathbb{N}$ as follows:*

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

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. (1) By induction on the number m such that $\mathcal{U}_{g_0}^{(m)} = f$. If $f = \mathcal{U}_{g_0}^{(0)} = 0^{\alpha-1}$, then $g_0 * f = g_0 * 0^{\alpha-1}$ is $\mathcal{U}$-generated by hypothesis.
If

$$f = \mathcal{U}_{g_0}^{(k+1)} = \mathcal{U}_{g_0}^{(k)} \oplus \mathcal{U}_{g_0}(\mathcal{U}_{g_0}^{(k)})$$

then $g_0 * \mathcal{U}_{g_0}^{(k)}$ is $\mathcal{U}$-generated by inductive hypothesis, i.e. for some n, $g_0 * \mathcal{U}_{g_0}^{(k)} = \mathcal{U}^{(n)}$. We have two cases:

i) $\mathcal{U}_{g_0}(\mathcal{U}_{g_0}^{(k)}) = \langle \gamma, x, z\rangle$. Then, by definition of $\mathcal{U}_{g_0}$

$$\mathcal{U}(g_0 * \mathcal{U}_{g_0}^{(k)}) = \langle \gamma + 1, x, z\rangle$$

Therefore,

$$\begin{aligned} \mathcal{U}^{(n+1)} &= \mathcal{U}^{(n)} \oplus \mathcal{U}(\mathcal{U}^{(n)}) \\ &= (g_0 * \mathcal{U}_{g_0}^{(k)}) \oplus \mathcal{U}(g_0 * \mathcal{U}_{g_0}^{(k)}) \\ &= (g_0 * \mathcal{U}_{g_0}^{(k)}) \oplus \langle \gamma + 1, x, z\rangle \\ &= g_0 * (\mathcal{U}_{g_0}^{(k)} \oplus \langle \gamma, x, z\rangle) \\ &= g_0 * f \end{aligned}$$

ii) $\mathcal{U}_{g_0}(\mathcal{U}_{g_0}^{(k+1)}) = \emptyset$. Then

$$f = \mathcal{U}_{g_0}^{(k)} \oplus \emptyset = \mathcal{U}_{g_0}^{(k)}$$

Hence $g_0 * f$ is $\mathcal{U}$-generated by induction hypothesis.

[2]As in chapter 4, $g_0 \leq g$ iff for all x $g_0(x) \neq 0 \implies g_0(x) = g(x)$

(2) Let f be a $\mathcal{U}_g$-generated finite zero of $\mathcal{U}_g$. By definition of g, for every m such that, for some numbers x, y,

$$\mathcal{U}(\mathcal{U}^{(m)}) = \langle 0, x, y\rangle$$

we have that $\mathcal{U}^{(m+1)} = h_m * 0^{(\alpha-1)}$ for some finite function $h_m \leq g$. $h_m * 0^{(\alpha-1)}$ is $\mathcal{U}$-generated by definition. The sequence of all h_m is increasing, and, by definition of g, the limit of all h_m is indeed g. Thus, for every finite function $h \leq g$ we have $h \leq h_m \leq g$ for some finite h_m such that $h_m * 0^{(\alpha-1)}$ is $\mathcal{U}$-generated.

By assumption f is $\mathcal{U}_g$-generated, that is, $f = \mathcal{U}_g^{(n)}$ for some n. By continuity of $\mathcal{U}(g * f)$ in g, and by continuity of $\mathcal{U}_g^{(m)}$ in g for any $m \leq n$, we deduce that there is some finite function $h \leq g$ such that for all functions $h \leq g_0 \leq g$ the two conditions below hold:

$$\mathcal{U}(g_0 * f) = \mathcal{U}(g * f)$$

$$f = \mathcal{U}_{g_0}^{(n)}$$

that is, f is an $\mathcal{U}_{g_0}$-generated finite zero of $\mathcal{U}_{g_0}$. By the discussion above, we may choose some finite g_0 such that $h \leq g_0 \leq g$ and $g_0 * 0^{(\alpha-1)}$ is $\mathcal{U}$-generated. By point (1), $g_0 * f$ is $\mathcal{U}$-generated:

$$g_0 * f = \mathcal{U}^{(n)} \tag{6.2}$$

for some n. Suppose

$$\mathcal{U}(g_0 * f) = \langle 0, x, z\rangle \tag{6.3}$$

for some x, z: we show it is impossible and hence obtain that $\mathcal{U}(g_0 * f) = \emptyset$, by the fact that $\mathcal{U}_{g_0}(f) = \emptyset$ and definition of $\mathcal{U}_{g_0}$. Combining (6.2) and (6.3), we obtain

$$\mathcal{U}(\mathcal{U}^{(n)}) = \langle 0, x, z\rangle$$

By definition of g, for some $m \leq n$

$$\mathcal{U}(\mathcal{U}^{(m)}) = \langle 0, x, y\rangle \wedge g(x) = y$$

This last fact plus (6.3) imply that $x \neq x$, since by definition $\mathcal{U}$ is an update procedure: impossible.

□

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

THEOREM 6.3.6 (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. We define, by transfinite induction, a function $g : \alpha \to (\mathbb{N} \to \mathbb{N})$ as follows. Suppose we have defined $g_\gamma : \mathbb{N} \to \mathbb{N}$, for every $\gamma < \beta < \alpha$. Define the sequence $g_{<\beta} : \beta \to (\mathbb{N} \to \mathbb{N})$ of them all

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

Then define

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

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

$$g := g_{<\alpha}$$

We first want to show that there exists a β such that $\mathcal{U}_{g_{<\beta}}$ has a $\mathcal{U}_{g_{<\beta}}$-generated finite zero. We have two cases:

(1) α is a successor ordinal. Then, by fact 1, $\mathcal{U}_{g<\alpha}$ is an update procedure of ordinal 1, which has a $\mathcal{U}_{g<\alpha}$-generated finite zero (see chapter 4).

(2) α is a limit ordinal. Then, by continuity of $\mathcal{U}$, there is some $\beta < \alpha$ such that for all $\beta \leq \delta < \alpha$

$$\mathcal{U}(g) = \mathcal{U}(g_{<\delta} * 0^{(\alpha-\delta)})$$

If $\mathcal{U}(g) = \emptyset$, then by definition 6.3.3

$$\mathcal{U}_{g_{<\beta}}(0^{(\alpha-\beta)}) = \emptyset$$

and we are done. If $\mathcal{U}(g) = \langle \gamma, n, m \rangle$, without loss of generality we can assume we have chosen β such that $\gamma < \beta$. Again by definition 6.3.3 of $\mathcal{U}_{g_{<\beta}}$

$$\mathcal{U}_{g_{<\beta}}(0^{(\alpha-\beta)}) = \emptyset$$

and we are done.

Let now

$$\beta_0 := \mathsf{min}\{\beta \mid \mathcal{U}_{g_{<\beta}} \text{ has a } \mathcal{U}_{g_{<\beta}}\text{-generated finite zero}\}$$

β_0 cannot be a successor, otherwise if we let $\beta_0 = \beta_1 + 1$

$$g_{<\beta_0} = g_{<(\beta_1)} * g_{\beta_1}$$

and hence by fact 1 point (2)

$$\mathcal{U}_{g_{<\beta_0}} = (\mathcal{U}_{g_{<\beta_1}})_{g_{\beta_1}}$$

and by reduction lemma 6.3.5 $\mathcal{U}_{g_{<\beta_1}}$ would have a $\mathcal{U}_{g_{<\beta_1}}$-generated finite zero. But β_0 also cannot be a limit ordinal, otherwise by reduction lemma 6.3.4, for some $\gamma < \beta_0$, $\mathcal{U}_{g_{<\gamma}}$ would have a $\mathcal{U}_{g_{<\gamma}}$-generated finite zero. We conclude that $\beta_0 = 0$. Since $\mathcal{U}_{g<0} = \mathcal{U}$, we obtain the thesis. □

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

Zeros of transfinite 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.

Definition 6.4.1 (Bar Recursion Operator, Spector's System B, Type Level of Bar Recursion). In the following, we will work with Spector's system B which is Gödel's T augmented with constants $\mathsf{BR}_{\tau,\sigma}, \Psi_{\tau,\sigma}$ respectively of type

$$T_1 \to T_2 \to T_3 \to T_4 \to \tau$$

and

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

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

$$\mathsf{BR}_{\tau,\sigma} YGHs \stackrel{\tau}{=} \begin{cases} Gs & \text{if } Y\hat{s} < |s| \\ Hs(\lambda x^\sigma \mathsf{BR}_{\tau,\sigma} YGH(s * x)) & \text{otherwise} \end{cases} \tag{6.4}$$

where $s * x$ denotes the finite sequence s followed by x, $\hat{s}$ denote 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 6.4 holds, we have to add to system B the following reduction rules (see Berger [13]):

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

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. [36])

Definition 6.4.2 (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 $|\emptyset| = \langle 0, \ldots, 0 \rangle$.

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 6.2.1, 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.

6.5. 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 6.5.1 (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 [39]. 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 6.5.1. 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 satisfied condition). We let σ as 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:

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

with

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

In particular

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

Now, we have two cases:

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

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

and so $\sigma(n) = l$, which is impossible, by definition 6.2.1 of update procedure, point (2), for $\mathcal{U}\sigma = |(n,m)|$.

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

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

Therefore

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

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 6.5.2 (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 6.5.1

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

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

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

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

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

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

$$\mathcal{U}(\mathsf{BR}(s * g_1)) = |(|s|, n, m)|$$

and

$$\mathcal{U}(\mathsf{BR}(s * g_2)) = |(|s|, h, l)|$$

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 6.5.1, 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 6.5.3 (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. By induction on k. The cases $k = 0, 1$ have already been taken care. Now, we want to prove the thesis for $k+1$, with $k > 0$. The finite function $\sigma : [\omega^{k+1} \to (\mathbb{N} \to \mathbb{N})]$ 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, with each $\sigma(i)$ of type $[\omega^k \to (\mathbb{N} \to \mathbb{N})]$. In the following s represents a sequence of functions of type $[\omega^k \to (\mathbb{N} \to \mathbb{N})]$. We recall that an ordinal less than ω^{k+1} is coded as a pair (γ, β), with $\gamma \in \mathbb{N}$ and $\beta \in \mathbb{N}^k$. We again present the construction gradually. To begin with, suppose we are able to define - uniformly on s - terms $\mathsf{BR}(s)$ and $g_s : [\omega^k \to (\mathbb{N} \to \mathbb{N})]$ satisfying the following equation for every s:

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

Let

$$\sigma := \mathsf{Zero}_{\omega^{k+1}}(\mathcal{U}) := \mathsf{BR}(\langle\rangle)$$

We prove that σ is a finite zero of $\mathcal{U}$. We show this by proving that $\mathcal{U}\sigma = ((\gamma, \beta), n, m)$ is impossible. As in the proof of theorem 6.5.1

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

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

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

we obtain that $\mathcal{U}\sigma \neq ((\gamma, \beta), n, m)$: 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

$$g_s := \mathsf{Zero}_{\omega^k}(\lambda f^{[\omega^k \to (\mathbb{N}\to\mathbb{N})]} \mathcal{U}_{|s|}(\mathsf{BR}(s * f)))$$

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

$$\mathcal{U}_i := \lambda f^{[\omega^{k+1} \to (\mathbb{N}\to\mathbb{N})]}.\ \mathsf{if}\ \mathcal{U}(f) = ((i, \delta), n, m)\ \mathsf{then}\ (\delta, n, m)\ \mathsf{else}\ \emptyset$$

We prove now that in fact $\mathcal{U}(\mathsf{BR}(s * g_s)) \neq |((|s|, \beta), n, m)|$ for all β, n, m. First, observe 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. First, we want to show that

$$F_s := \lambda f^{[\omega^k \to (\mathbb{N}\to\mathbb{N})]} \mathcal{U}_{|s|}(\mathsf{BR}(s * f))$$

is an update procedure of ordinal ω^k. Suppose for some δ of type $[\omega^k]$: $F_s g_1 = |(\delta, n, m)|$, $\forall \delta_0 < \delta.\ (g_1)_{\delta_0} = (g_2)_{\delta_0}$, $(g_2)_\delta(n) = m$ and $F_s g_2 = |(\delta, h, l)|$. Then, by definition of F_s, it must be that

$$\mathcal{U}(\mathsf{BR}(s * g_1)) = |((|s|, \delta), n, m)|$$

and

$$\mathcal{U}(\mathsf{BR}(s * g_2)) = |((|s|, \delta), h, l)|$$

Since $\mathcal{U}$ is an update procedure and

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

then $h \neq n$ must hold; therefore F_s is an update procedure of ordinal ω^k. But by definition of g_s and induction hypothesis, this means that

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

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

□

6.6. Case Study: Elementary Analysis

In this section, we give a three-step description of the epsilon substitution method for Elementary Analysis. Every step corresponds to one of the three stages in which - according to section 6.2 - learning based computational interpretations of predicative classical Arithmetic can be decomposed. As main foundational result one obtains a constructive proof that the zero theorem for update procedures of ordinal less that $\omega \cdot 2$ implies the 1-consistency of Elementary Analysis. More precisely, any zero of such an update procedure can be used to compute witnesses for Π^2_0 formulas.

The content of this section is based on Mints et al. [35] and may be considered as an informal survey and a general guide to the reading of the epsilon substitution method in the light of our ideas on learning. Neither full details nor full proofs will be provided, but our description should be clear enough for the reader to gain an understanding of the basic ideas underpinning the epsilon method and its learning based interpretation.

We first define the language of Elementary Analysis, which is a fragment of second order Arithmetic in which second order quantification ranges over arithmetical formulas (possibly with free set variables).

Definition 6.6.1 (Language $\mathcal{L}_{\mathsf{EA}}$ of EA). The *terms* of $\mathcal{L}_{\mathsf{EA}}$ are inductively defined as follow:

(1) Numerical variables $x, y, z, \ldots$ are terms of type 0.

(2) Set variables $X, Y, Z, \ldots$ are terms of type 1.

(3) $\mathsf{0}$ is a term of type 0.

(4) If t is a term of type 0, $\mathsf{S}(t)$ is term of type 0.

The *formulas* of $\mathcal{L}_{\mathsf{EA}}$ are inductively defined as follows:

(1) For every natural number n, there is a denumerable set of n-ary predicate constants, one for every computable predicate over n-uples of natural numbers. If P is a n-ary predicate constant and $t_1, \ldots, t_n$ are terms of type 0, then $Pt_1 \ldots t_n$ is an atomic formula.

(2) If t is a term of type 0 and X a variable of type 1, then $t \in X$ is an atomic formula.

(3) If A and B are formulas, then $A \wedge B$, $A \to B$, $\neg A$ are formulas.

(4) If A is a formula and v is a variable, $\exists v A$ is a formula and $\forall v A$ is defined as $\neg\exists v \neg A$.

If A is a formula and z a variable of type 0 free in A, then $\lambda z A$ is a *lambda set*; $\lambda z A$ is said to be *arithmetical* if it contains no bound set variables. The formula $B(\lambda z A/X)$ is defined as the formula obtained from B by substituting each atomic formula $t \in X$ of B with $A(t/z)$, as usual without capture of variables.

We now define the axioms and inference rules of EA.

Definition 6.6.2 (Axioms and Inference Rules of EA). The *axioms* of EA are formulas of $\mathcal{L}_{\mathsf{EA}}$ defined as follows:

(1) Propositional tautologies are axioms.

(2) Definitions of predicate constants are axioms, e.g. for *add* predicate

$$add(x, 0, x) \text{ and } add(x, y, z) \to add(x, \mathsf{S}(y), \mathsf{S}(z))$$

(3) $x = x$ and $x = y \to A(x) \to A(y)$ are equality axioms.

(4) $\neg \mathsf{S}(x) = 0$ and $\mathsf{S}(x) = \mathsf{S}(y) \to x = y$ are axioms.

(5) $A(0) \to (\forall x.A(x) \to A(\mathsf{S}(x))) \to \forall x A(x)$ is the induction axiom scheme.

(6) $A(t/x) \to \exists x A$ is an axiom for every term t of type 0.

(7) $A(T/X) \to \exists X A$ is an axiom if T is a set variable or an arithmetical lambda set.

The *inference rules* of EA are modus ponens

$$\frac{A \to B \quad A}{B}$$

and

$$\frac{A \to C}{\exists v A \to C}$$

with the standard proviso that v does not occur free in C.

We are now ready to take the first step of a learning based interpretation.

6.6.1. First Stage: Identification of a Sequence of non Computable Functions F. We now define the sequence of non computable functions needed to give a computational interpretation of EA. We do that by first introducing the concept of epsilon term.

Definition 6.6.3 (Language $\mathcal{L}_{\mathsf{EA}\epsilon}$). We define by simultaneous induction the *terms* and the *formulas* of $\mathcal{L}_{\mathsf{EA}\epsilon}$:

(1) Numerical variables $x, y, z, \ldots$ are terms of type 0.

(2) Set variables $X, Y, Z, \ldots$ are terms of type 1.

(3) $\mathsf{0}$ is a term of type 0.

(4) If t is a term of type 0, $\mathsf{S}(t)$ is term of type 0.

(5) For every n-ary predicate constant P of $\mathcal{L}_{\mathsf{EA}}$, if $t_1, \ldots, t_n$ are terms of type 0, then $Pt_1 \ldots t_n$ is an atomic formula.

(6) If t is a term of type 0 and T a term of type 1, then $t \in T$ is an atomic formula.

(7) If A and B are formulas, then $A \wedge B$, $A \to B$, $\neg A$ are formulas.

(8) If A is a formula and v is a variable, then ϵvA is an *epsilon term* (of type equal to the type of v) and v is considered bound in ϵvA.

If A is a formula and z a variable of type 0 free in A, then λzA is a *lambda term* (but we do not ask it is in $\mathcal{L}_{\mathsf{EA}\epsilon}$). A formula or a lambda term or a term is said to be an *expression of* $\mathsf{EA}\epsilon$ and is *arithmetical* if it contains no bound set variables and *canonical* if it is closed (i.e. no free variables occurs in it) and does not have closed epsilon terms as subterms. λzA is *regular* if it is of the form $(\lambda zB)[t_1/v_1 \dots t_n/v_n]$ with λzB arithmetical and $t_1, \dots, t_n$ any terms.

Canonical epsilon terms are the ones that are assigned a meaning. The intended denotation of a canonical epsilon term ϵxA is the least number n such that $A(n)$ is true while the denotation of ϵXA is an arithmetical canonical lambda term λzG (which represents an arithmetical set) such that $A(\lambda zG/X)$ is true. In other words, the following *critical* formulas should be true:

$$A(t/x) \to A(\epsilon xA/x)$$

$$A(T/X) \to A(\epsilon XA/X)$$

where t is any term of type 0 and T is an epsilon term of type 1 or a regular lambda term. The notion of truth for formulas of $\mathsf{EA}\epsilon$ requires further explanation: in order to evaluate their truth, first epsilon terms must be evaluated and hence eliminated.

Definition 6.6.4 (Substitutions, Evaluations of Epsilon Terms). We define:

(1) An *epsilon substitution* S is a function from the set of canonical epsilon terms to the set of numerals and arithmetical canonical lambda terms such that $S(\epsilon xA)$ is always a numeral and $S(\epsilon XA)$ is always an arithmetical canonical lambda term.

(2) Let t_1, t_2 be expressions of $\mathsf{EA}\epsilon$ and S an epsilon substitution. We write $t_1 \mapsto_S t_2$ if t_2 is obtained from t_1 either by substituting one of its canonical epsilon subterms ϵxA with $S(\epsilon xA)$ or replacing one of its subformulas $t \in \epsilon XA$ with $G(t)$, where $S(\epsilon XA) = \lambda zG$.

(3) An expression t of $\mathsf{EA}\epsilon$ is said to be in S-normal form if there is no t_1 such that $t \mapsto_S t_1$. We indicate with $|t|_S$ the unique S-normal form of t, which exists by theorem 6.6.1 below.

The truth value of a closed formula A of $\mathsf{EA}\epsilon$ is the truth value of $|A|_S$ (which does not contain epsilon terms): so it is always relative to a epsilon substitution S.

The relation $\mapsto_S$ is well founded and has Church-Rosser property (see Mints et al. [35]).

Theorem 6.6.1 (Normalization and Church-Rosser). *For every S, the relation $\mapsto_S$ is well founded and every expression t of $\mathsf{EA}\epsilon$ has an S-normal form.*

We now need to measure the "computational strength" of an expression t of $\mathsf{EA}\epsilon$. Intuitively, from the computational point of view, an epsilon term ϵxA represent a recursion theoretic jump, because in general one has to enumerate all natural numbers in order to decide if an n exists such that $|A(n)|_S$ is true. So, closed arithmetical expressions will have a computational strength below ω, because no more than a finite number of jumps is done inside them by their epsilon subterms. Instead, an epsilon term ϵXA must have a computational strength of at least ω because one has to know all the values of arithmetical canonical epsilon terms in the first place if he wants to determine whether there exists a canonical arithmetical lambda term λzG such that $|A(\lambda zG/X)|_S$ is true. Indeed one can assign to expressions a computational strength which is always less that $\omega \cdot 2$. This is done through the so called *rank function*, which we introduce only by exposing the properties that it must have (for the actual definition and details, see Mints et al. [35]).

THEOREM 6.6.2 (RANK FUNCTION). *There exists a function* rk *from the set of expressions of* $\mathsf{EA}\epsilon$ *to* $\omega \cdot 2$ *such that the following holds. For every epsilon substitution* S *and ordinal* α*, denote with* $S_{<\alpha}$ *the function mapping* e *to* $S(e)$ *if* $\mathsf{rk}(e) \leq \alpha$*, to* 0 *or* $0^1 := (\lambda z.z = z)$ *otherwise (according to the type of* e*); then*

(1) *For every canonical epsilon term* ϵvA

$$\mathsf{rk}(\epsilon vA) > \mathsf{rk}(A(e/v))$$

whenever e *is a numeral or an arithmetical canonical lambda term.*

(2) *For every expression* e *of* $\mathsf{EA}\epsilon$ *and epsilon substitutions* S_1, S_2*, if* $\mathsf{rk}(e) = \alpha$ *and* $(S_1)_{<\alpha} = (S_2)_{<\alpha}$*, then* $|e|_{S_1} = |e|_{S_2}$*.*

The above theorem 6.6.2 is crucial and its meaning is the following. Given any canonical epsilon term ϵxA, any substitution S and natural number $n = S(\epsilon xA)$, in order to check whether n is a correct denotation for ϵxA, we have to determine the truth value of

$$|A(n/x)|_S$$

Since $\mathsf{rk}(\epsilon xA)$ is strictly greater than $\mathsf{rk}(A(n/x))$, the truth of $|A(n/x)|_S$ depends only on the values that S assigns to epsilon terms of rank strictly *less* than that of ϵxA. So the meaning of ϵxA is predicatively determined by the meaning of epsilon terms of lower rank. The same holds for canonical epsilon terms of type 1.

We are now able to define the sequence of functions that will enable us to define classical witness for formulas in EA and that we shall try to approximate. Suppose that for every ordinal $\alpha < 2 \cdot \omega$ we have a primitive recursive enumeration $\epsilon_0^\alpha, \epsilon_1^\alpha, \ldots$ of canonical epsilon terms of rank equal to α and a primitive recursive enumeration $\lambda_0, \lambda_1, \ldots$, of canonical arithmetical lambda terms. We associate to any function $f : \omega \cdot 2 \to (\mathbb{N} \to \mathbb{N})$ the epsilon substitution S_f such that:

$$f_\alpha(n) = \begin{cases} m & \text{if } S_f(\epsilon_n^\alpha) = m \wedge \epsilon_n^\alpha = \epsilon xA \\ l & \text{if } S_f(\epsilon_n^\alpha) = \lambda_l \wedge \epsilon_n^\alpha = \epsilon XA \end{cases}$$

It is easy to see - using classical logic - that there exists an epsilon substitution S which makes true every critical formula C, i.e. $|C|_S$ is true: just start by assigning values to canonical epsilon terms of rank 1, then to those of rank 2 and so on. Then our target collection $F : \omega \cdot 2 \to (\mathbb{N} \to \mathbb{N})$ of functions is the one such that $\mathsf{S} = S_F$.

6.6.2. Second Stage: Definition of Classical Witnesses by Programs Recursive in F. Using epsilon terms one can define classical witness for any provable formula of EA by first translating formulas of EA into formulas of $\mathsf{EA}\epsilon$, using the equivalence

$$\exists v A \equiv A(\epsilon v A/v)$$

Definition 6.6.5 (Translation of Formulas of EA into Formulas $\mathsf{EA}\epsilon$). We define a translation of Formulas of EA into Formulas $\mathsf{EA}\epsilon$ by induction as follows:

(1) If P is atomic, $P^* := P$.

(2) $(\neg A)^* := \neg A^*$.

(3) $(A \wedge B)^* := A^* \wedge B^*$.

(4) $(A \to B)^* := A^* \to B^*$.

(5) $(\exists v A)^* := A^*(\epsilon v A^*/v)$

We now define the axioms and inference rules for $\mathsf{EA}\epsilon$.

Definition 6.6.6 (Axioms and Inference Rules of $\mathsf{EA}\epsilon$). The *axioms* of $\mathsf{EA}\epsilon$ are formulas of $\mathcal{L}_{\mathsf{EA}\epsilon}$ defined as follows:

(1) Propositional tautologies are axioms.

(2) Definitions of predicate constants are axioms, e.g. for *add* predicate

$$add(x, 0, x) \text{ and } add(x, y, z) \to add(x, \mathsf{S}(y), \mathsf{S}(z))$$

(3) $x = x$ and $x = y \to A(x) \to A(y)$ are equality axioms.

(4) $\neg \mathsf{S}(x) = 0$ and $\mathsf{S}(x) = \mathsf{S}(y) \to x = y$ are axioms.

(5) Minimality axioms: $\epsilon x A = \mathsf{S}(t) \to \neg A(t)$

(6) Critical formulas:

$$\neg s = 0 \to s = \mathsf{S}(\epsilon x s = \mathsf{S}(x))$$

$$A(t/x) \to A(\epsilon x A/x)$$

$$A(T/X) \to A(\epsilon X A/X)$$

where t is any term of type 0 and T is either a term of type 1 or a regular lambda term.

The only *inference rule* of $\mathsf{EA}\epsilon$ is modus ponens.

The following theorem shows that EA can be embedded in the quantifier free system $\mathsf{EA}\epsilon$. This allows one to extract witnesses for existential statements provable in EA. As usual $\vdash$ denotes provability.

THEOREM 6.6.3 (CLASSICAL WITNESSES FOR PROVABLE FORMULAS OF EA). *The following holds:*

(1) *Suppose that* $\mathsf{EA} \vdash A$. *Then* $\mathsf{EA}\epsilon \vdash A^*$.

(2) *Suppose that* $\mathsf{EA} \vdash \exists x A$, *with* A *atomic. Then there exists a finite sequence* $C_1, \ldots, C_n$ *of closed critical formulas of* $\mathsf{EA}\epsilon$ *and a closed term* t *of* $\mathsf{EA}\epsilon$ *such that if for all* i, $|C_i|_S$ *is true, then* $|t|_S = n$ *and* $A(n)$ *is true.*

By the above theorem 6.6.3, it is now clear that witnesses for EA can be computed by programs recursive in F, because F represent an epsilon substitution S_F which makes every critical formula true.

6.6.3. Stage Three: Learning Processes Approximating F. In order to compute witnesses for EA is necessary to find a good finite approximation of F, in order to satisfy some finite set of critical formulas. This is the point when update procedures come into the scene. The fundamental property of a closed critical formula C, for example of the form

$$A(t/x) \to A(\epsilon x A/x)$$

is that from the fact $|C|_S$ is false one can always learn something. Suppose $|C|_S$ is false. In this case, if $S(\epsilon x|A|_S) = m$, one has that $|A(m/x)|_S$ is false. Fortunately, if $|t|_S = n$, the formula $|A(n/x)|_S$ is true since $|C|_S$ is false and so $|A(n_0/x)|_S$ is true for some minimal $n_0 \leq n$. So one learns a new value n_0 that can be assigned to $\epsilon x|A|_S$. Analogously, if C is a closed critical formula of the form

$$A(T/X) \to A(\epsilon X A/X)$$

and $|C|_S$ is false, if we suppose $S(\epsilon X|A|_S) = \lambda z G$, one has that $|A(\lambda z G/X)|_S$ is false. Fortunately, if $|T|_S = \lambda z H$, the formula $|A(\lambda z H/X)|_S$ happens to be true. So one learns a new value $\lambda z H$ that can be assigned to $\epsilon X|A|_S$. Observe that is not a priori obvious that $\lambda z H$ is an arithmetical lambda term, but in fact this can be proved given the assumptions we have made on T (again, for details see Mints et al. [35]).

From now on, fix a finite sequence of critical formulas $C_0, \ldots, C_N$ (for brevity only of the two forms considered above). We want to define an update procedure of ordinal $\omega + k$ out of it (with $k \in \omega$), any of whose finite zeros will represent an epsilon substitution making all the critical formulas true.

DEFINITION 6.6.7 (UPDATE PROCEDURE FOR $C_0, \ldots, C_N$). We define an update procedure $\mathcal{U}$ of ordinal $\omega + k$. Let $f : \omega + k \to (\mathbb{N} \to \mathbb{N})$. If for every i, $|C_i|_{S_f}$ is true, we set

$$\mathcal{U}(f) = \emptyset$$

Otherwise, consider the first i such that $|C_i|_{S_f}$ is false. If

$$C_i = A(t/x) \to A(\epsilon x A/x)$$

and $|t|_{S_f} = m$ and $\epsilon x|A|_{S_f} = \epsilon^{\alpha}_n$, we set

$$\mathcal{U}(f) = \langle \alpha, n, m_0 \rangle$$

where $m_0 \leq m$ is the smallest among the i such that $|A(i/x)|_{S_f}$ is true (which exists since $|A(m/x)|_{S_f}$ is true). If

$$C_i = A(T/X) \to A(\epsilon X A/X)$$

and $|T|_{S_f} = \lambda zH = \lambda_m$ and $\epsilon X|A|_{S_f} = \epsilon^\alpha_n$, we set

$$\mathcal{U}(f) = \langle \alpha, n, m\rangle$$

We now show that $\mathcal{U}$ is a well defined update procedure.

THEOREM 6.6.4 (ADEQUACY OF $\mathcal{U}$). *$\mathcal{U}$ is an update procedure of ordinal $\omega + k$, for some $k \in \omega$.*

PROOF. We skip the proof that $\mathcal{U}$ is well defined, which amounts to show that for every substitution S, if $\mathsf{rk}(e) \geq \omega$, then $\mathsf{rk}(|e|_S) \leq \mathsf{rk}(e)$: this ensures an upper bound on the rank of the terms that are evaluated by $\mathcal{U}$ in its computations and so $\mathcal{U}$ never updates values for epsilon terms of rank greater than $\omega + k$, if k is chosen large enough.
We prove instead that $\mathcal{U}$ is an update procedure. $\mathcal{U}$ is continuous, since for every f only a finite number of values of S_f and hence of f are used to compute $\mathcal{U}(f)$. Suppose now that $\mathcal{U}(f) = \langle \beta, n, m\rangle$, for all $\gamma < \beta$ $f_\gamma = g_\gamma$, $g_\beta(n) = m$ and $\mathcal{U}(g) = \langle \beta, h, l\rangle$. Suppose $h = n$: we have to prove it is impossible. Consider the first i such that $|C_i|_{S_f}$ is false. Suppose C_i is of the form

$$C_i = A(t/x) \to A(\epsilon x A/x)$$

Then by definition of $\mathcal{U}(f)$, $\epsilon x|A|_{S_f} = \epsilon^\beta_n$ and m is the smallest among the i such that $|A(i/x)|_{S_f}$ is true. Furthermore, consider the first j such that $|C_j|_{S_g}$ is false. By definition of $\mathcal{U}(g)$ and since $h = n$ we have

$$C_j = B(t/v) \to B(\epsilon v B/v)$$

with $\epsilon v|B|_{S_g} = \epsilon^\beta_n$. Therefore, $|A|_{S_f} = |B|_{S_g}$. Moreover, let

$$\delta := \mathsf{rk}(|B|_{S_g}(m/x))$$

By theorem 6.6.2, point (1), $\delta < \beta = \mathsf{rk}(\epsilon x|B|_{S_g})$. By hypothesis

$$(S_g)_{\leq\delta} = (S_f)_{\leq\delta}$$

So by theorem 6.6.2, point (2)

$$|A(m/x)|_{S_f} = ||A|_{S_f}(m/x)|_{S_f} = ||B|_{S_g}(m/x)|_{S_f} = ||B|_{S_g}(m/x)|_{S_g} = |B(m/x)|_{S_g}$$

Since $g_\beta(n) = m$, we have

$$S_g(\epsilon v|B|_{S_g}) = S_g(\epsilon^\beta_n) = m$$

Thus

$$|B(\epsilon v B/v)|_{S_g} = |B(m/x)|_{S_g}$$

But $|A(m/x)|_{S_f}$ is true by construction and so $|B(\epsilon v B/v)|_{S_g}$ itself must be true, which contradicts the assumption that $|C_j|_{S_g}$ is false.
An analogous reasoning yields a contradiction when C_i is of the form

$$A(T/x) \to A(\epsilon X A/X)$$

□

THEOREM 6.6.1 (1-CONSISTENCY OF ELEMENTARY ANALYSIS). *If for all $k \in \omega$ every update procedure of ordinal $\omega + k$ has a finite zero, then Elementary Analysis is 1-consistent.*

PROOF. By theorem 6.6.3, it is enough to show that, given any finite sequence of critical formulas ($C_1, \ldots, C_N$ without loss of generality), there exists a finite epsilon substitution that makes true every formula. This amounts to show that $\mathcal{U}$ has a finite zero, which is true by theorem 6.6.4 and hypothesis. □

6.7. Further Work

Much remains to be done and we plan to address the following issues in the future.

Our constructive proof of the zero theorem for typed update procedures of ordinal ω^k is not optimal, in the sense that, by Howard [30], it should be possible to use only system T plus bar recursion of type 0.

Moreover, a more self contained proof that the zero theorem for update procedures of ordinal less that $\omega \cdot 2$ implies the consistency of EA is a major aim.

Bibliography

[1] Y. Akama, S. Berardi, S. Hayashi, U. Kohlenbach, *An Arithmetical Hierarchy of the Law of Excluded Middle and Related Principles*, in: LICS 2004, pp. 192-201.

[2] W. Ackermann, *Zur Widerspruchsfreiheit der Zahlentheorie*, Mathematische Annalen, 117, pp. 162194 (1940)

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

[4] F. Aschieri, *Interactive Learning Based Realizability and 1-Backtracking Games*, Proceedings of Classical Logic and Computation, to appear in Electronic Proceedings in Theoretical Computer Science

[5] J. Avigad, *Update Procedures and 1-Consistency of Arithmetic*, Mathematical Logic Quarterly, volume 48, 2002.

[6] S. Berardi, *Personal Communication*, 2009

[7] S. Berardi, *Classical Logic as Limit Completion*, MSCS, Vol. 15, n.1, 2005, pp.167-200.

[8] S. Berardi, *Some intuitionistic equivalents of classical principles for degree 2 formulas*, Annals of Pure and Applied Logic, Vol. 139, n.1-3, 2006, pp.185-200.

[9] S. Berardi, T. Coquand, S. Hayashi, *Games with 1-Bactracking*, Annals of Pure and Applied Logic, 2010.

[10] S. Berardi, U. de' Liguoro, *A calculus of realizers for* EM_1*-Arithmetic*, Proceedings of Computer Science Logic 2008, in LNCS 5213, pag 215-229 (2008)

[11] S. Berardi and U. de' Liguoro, *Toward the interpretation of non-constructive reasoning as non-monotonic learning*, Information and Computation, vol. 207, 1, pag. 63-81, (2009).

[12] S. Berardi and U. de' Liguoro, *Interactive Realizers and Monads*, Submitted to TOCL, 2010.

`http://www.di.unito.it/~deligu/papers/InteractiveRealizers.pdf`

[13] U. Berger, *Continuous Semantics for Strong Normalization*, Lecture Notes in Computer Science 3526, 23–34, 2005

[14] T. Coquand, *A Semantic of Evidence for Classical Arithmetic*, Journal of Symbolic Logic 60, pag 325-337 (1995)

[15] D. v. Dalen, *Logic and Structure*, Springer-Verlag, 3^{rd} Ed., Berlin Heidelberg (1994)

[16] M. Escardo, P. Oliva, *Selection Functions, Bar Recursion, and Backward Induction*, Mathematical Structures in Computer Science, 2010.

[17] S. Feferman and W. Sieg, *Proof-theoretic equivalences between classical and constructive theories for analysis*, in: Buchholz et al. [1981], pp. 78142.

[18] W. Felscher, *Dialogues as a Foundation for Intuitionistic Logic*, Handbook of Philosophical Logic, 2nd Edition, Volume 5, pages 115-145 2002 Kluwer Academic

[19] G. Gentzen, *Die Widerspruchsfreiheit der reinen Zahlentheorie*. Mathematische Annalen, 112:493-565, 1935. English translation: The consistency of elementary number theory, in Szabo [465], pages 132-200.

[20] G. Gentzen, *Untersuchungen fiber das logische Schliessen. Mathematische Zeitschrift, 39:176-210, 405-431, 1935. English translation: Investigations into logical deduction, in Szabo [465], pages 68-131*

[21] J.-Y. Girard, *Proofs and Types*, Cambridge University Press (1989)

[22] E. M. Gold, *Limiting Recursion*, Journal of Symbolic Logic 30, pag. 28-48 (1965)

[23] Goldblatt, *Lectures on the Hyperreals*, Springer-Verlag, New York, 1998

[24] Nicolas D. Goodman, *Relativized Realizability in Intuitionistic Arithmetic of All Finite Types*,Journal of Symbolic Logic 43, 1, pag. 23-44 (1978).

[25] C. Gunter, *Semantics of Programmin Languages*, Mit Press, 1992

[26] S. Hayashi, R. Sumitomo, K. Shii, *Towards Animation of Proofs -Testing Proofs by Examples* - , Theoretical Computer Science (2002)

[27] S. Hayashi, *Can Proofs be Animated by Games?*, FI 77(4), pag 331-343 (2007)
[28] S. Hayashi, *Mathematics based on incremental learning - Excluded Middle and Inductive Inference*, Theoretical Computer Science 350, pag 125-139 (2006)
[29] J. Hintikka, G. Sandu *Game-Theoretical Semantics* in Handbook of Language and Computation, The MIT Press (1997)
[30] Howard, *Ordinal Analysis of Symple Cases of Bar Recursion*, The Journal of Symbolic Logic, volume 46, number 1, 1981
[31] S. C. Kleene, *On the Interpretation of Intuitionistic Number Theory*, Journal of Symbolic Logic 10(4), pag 109-124 (1945)
[32] U. Kohlenbach, *Applied Proof Theory*, Springer-Verlag, Berlin, Heidelberg, 2008
[33] G. Kreisel, *On the Interpretation of Non-Finitists Proofs*, The Journal of Symbolic Logic, vol. 17, 1952
[34] G. Kreisel, *Interpretation of analysis by means of constructive functionals of - nite types*, Heyting, A. (ed.), Constructivity in Mathematics, pp. 101128. North- Holland, Amsterdam (1959).
[35] G. Mints, S. Tupailo, W. Bucholz, *Epsilon Substitution Method for Elementary Analysis*, Archive for Mathematical Logic, volume 35, 1996
[36] G. Mints, S. Tupailo, *Epsilon Substitution Method for the Ramified Language and* Δ^1_1*-Comprehension Rule*, Logic and Foundations of Mathematics, 1999
[37] A. Miquel, *Relating classical realizability and negative translation for existential witness extraction.* In Typed Lambda Calculi and Applications (TLCA 2009), pp. 188-202, 2009
[38] P. Odifreddi, *Classical Recursion Theory*, Studies in Logic and Foundations of Mathematics, Elsevier, 1989
[39] P. Oliva, *Understanding and Using Spector's Bar Recursive Interpretation of Classical Analysis*, Proceedings of CiE'2006, LNCS 3988:423-434, Springer, 2006
[40] K. Popper, *The Logic of Scientific Discovery*, Routledge Classics, Routledge, London and New York (2002)
[41] H. Schwichtenberg, *On Bar Recursion of Type 0 and 1*, The Journal of Symbolic Logic, volume 44, number 3, 1979
[42] T. Skolem, *Selected Works in Logic* Universitets for laget, Oslo, 1970. Edited by Fenstad, J. E.
[43] M. H. Sorensen, P. Urzyczyn, *Lectures on the Curry-Howard isomorphism*, Studies in Logic and the Foundations of Mathematics, vol. 149, Elsevier, 2006