Interactive Realizability for Classical Peano Arithmetic with Skolem Axioms

Federico Aschieri

Laboratoire PPS, équipe PI.R2, Université Paris 7, INRIA & CNRS

Abstract

Interactive realizability is a computational semantics of classical Arithmetic. It is based on interactive learning and was originally designed to interpret excluded middle and Skolem axioms for simple existential formulas. A realizer represents a proof/construction depending on some state, which is an approximation of some Skolem functions. The realizer interacts with the environment, which may provide a counter-proof, a counterexample invalidating the current construction of the realizer. But the realizer is always able to turn such a negative outcome into a positive information, which consists in some new piece of knowledge learned about the mentioned Skolem functions. The aim of this work is to extend Interactive realizability to a system which includes classical first-order Peano Arithmetic with Skolem axioms. For witness extraction, the learning capabilities of realizers will be exploited according to the paradigm of learning by levels. In particular, realizers of atomic formulas will be update procedures in the sense of Avigad and thus will be understood as stratified-learning algorithms.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Interactive realizability, learning, classical Arithmetic, witness extraction

Digital Object Identifier 10.4230/LIPIcs.CSL.2012.31

1 Introduction

Do classical proofs have some constructive content? If yes, what is a construction in classical logic? On a first thought one is inclined to think that these questions cannot have any interesting answers in terms of effective computer programs.

Surely, a classical proof is a *mental construction*, for it is a succession of constructive steps interleaved with some ineffective considerations, which however appear to have a clear mental constructive significance. Indeed, when we use the excluded middle, we can clearly picture ourselves ideally deciding whether in a situation something holds or does not. When we use an axiom of comprehension, we employ a definite law in order to construct in our minds a perfectly determined collection of elements. When we use the axiom of choice, we may imagine ourselves to make arbitrary choices as long as it is needed.

Even if a classical proof seems a legitimate mental construction, it is still a long way to yield some effective computer program. Nevertheless, from the beginning of proof theory many results have been obtained in that direction, which clearly showed that classical proofs have a constructive content. Of course, we refer to the seminal results obtained by Hilbert's epsilon substitution method (see e.g [18]) and Gentzen's cut elimination [14]. Then, several other techniques have been introduced: among them, Gödel's double negation translation followed either by the Gödel functional interpretation [13] or Kreisel's modified realizability [16] and Friedman's translation [12]; finally, Curry-Howard correspondence (see e.g. [19]).

The Curry-Howard correspondence, first introduced for intuitionistic logic and finally extended to higher-order classical logic [17], clearly shows that a classical proof not only is a mental construction but has the very same syntactic structure of a program. In other words,

Computer Science Logic 2012 (CSL'12).
Editors: Patrick Cégielski, Arnaud Durand; pp. 31–45

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

a proof is an *effective program*. Thus the problem of explaining what is a construction in classical logic acquires a perfectly sound mathematical sense.

Though all these constructive interpretations may seem very different from each other, a deeper study shows that they are all based on the same concept: learning. As suggestively showed by Coquand [11], a classical proof yields a learning strategy in some class of games in which players can erase their moves and backtrack to earlier positions of the game.

Now, the most important problem to solve in order to *understand* and *implement efficiently* the constructive content of classical proofs, is to provide an accurate description of the nature and the structure of the knowledge that a program extracted from a classical proof gathers during its execution. Only after a precise description of learning has been completed, one can think of how to build efficient programs. This remains a complex task, but it is of central importance to start by defining a semantics for classical proofs explicitly based on learning.

In order to provide answers to those issues, Interactive realizability [1, 4, 6] has been developed: it is a realizability semantics based on states of knowledge and learning, designed for a system of Heyting Arithmetic with excluded middle and choice principles (Skolem axioms) restricted to Σ^0_1-formulas. In hindsight, it can be seen as modern evolution of the epsilon substitution method, refined and rebuilt around the Curry-Howard correspondence for classical logic and game semantics. The aim of this paper is to take the theory of Interactive realizability to the next level: we extend it in order to interpret a full classical system, which includes first-order Peano Arithmetic with Skolem axioms, that is, excluded middle, comprehension and choice over all first-order formulas.

The theory of Interactive realizability makes precise the intuitive considerations that we have made above. In particular, it explains how to interpret a classical proof, how to pass from the ideal *mental construction* it represents to a concrete *effective construction* (a program). The main concepts are indeed the following:

- $\mathcal{T}_{\text{Class}}$ *and mental constructions.* An interactive realizer is in the first place a term of $\mathcal{T}_{\text{Class}}$, a version of Gödel's system T enriched with Skolem function symbols for every arithmetical formulas. The terms of $\mathcal{T}_{\text{Class}}$ represent the mental constructions that one can obtain directly and intuitively from a classical proof.
- *States of Knowledge as Approximations.* Terms of $\mathcal{T}_{\text{Class}}$ are ineffective and, let to themselves, useless. Therefore, interactive realizers are always computed with respect to *states*, i.e. *approximations* of the Skolem functions they contain and thus effectiveness is recovered.
- *Learning.* Since finite approximations may be inadequate, results of computations may be wrong. But an interactive realizer is also a *self-correcting* program, able to spot incorrect values of the approximations used during computations and to correct them with *right* values. The new values are *learned*, remarkably, by realizers of classical principles and all the oracle values needed during each particular computation are acquired through learning processes. Here is the fundamental insight: classical principles may be computationally interpreted as learning devices.

All these ideas work very smoothly for the most simple instance of the excluded middle

$$\mathsf{EM}_1 := \forall x^{\mathbb{N}}.\ \exists y^{\mathbb{N}} P(x,y) \vee \forall y^{\mathbb{N}} \neg P(x,y) \ (P \text{ computable atomic predicate})$$

A realizer of this principle uses some approximation s of a Skolem function Φ for P in order to decide which side of EM_1 is true. If for some particular n, $P(n, s(n))$ holds, then the realizer has a witness for the left side; otherwise, it declares the right side to be true. However, if the environment asks the realizer for a construction of $\neg P(n,m)$ and actually m is a counterexample to its belief (i.e. $P(n,m)$ is true), then the realizer *corrects* the

approximation s as to output m on input n. We observe that this correction is sound on *absolute* grounds: m is a correct value for Φ on argument n.

In the case of a general instance of the excluded middle

$$\mathsf{EM} := \forall x^{\mathtt{N}}.\ \exists y^{\mathtt{N}} A(x,y) \vee \forall y^{\mathtt{N}} \neg A(x,y) \ (A \text{ first-order formula})$$

one stumbles across a serious problem: even if one has an approximation s of a Skolem function Φ for $A(x,y)$, how to compute whether $A(n,s(n))$ holds? In general, one cannot know for sure the truth value of a complex formula. A realizer thus may assume that $s(n)$ is not a witness and declare the right side to be true. However, a problem remains because learning is going to be by counterexamples: how to test whether an m given by a computational environment is such that $A(n,m)$ is true? The solution is to "compute" the truth value of $A(n,m)$ by using the approximations of the Skolem functions for the sub-formulas of A: for eliminating quantifiers, it suffices to use the equivalences $\exists y\, B(x,y) \equiv B(x,\psi(x))$, where ψ is a Skolem function for B. Since these approximations refer to Skolem functions for formulas of lower logical complexity than that of A, a realizer can determine on *relatively* firm grounds whether m is a correct value for Φ on argument n.

An important concept that we shall introduce is therefore that of *truth value of a formula in a state* and we shall study how it relates to realizability. Another novelty is that the self-corrections of realizers will not be absolute, but relative, and learning will be by *levels*. By this we mean that whenever a realizer gains some knowledge concerning a Skolem function for some formula, it may falsify a knowledge about another Skolem function for a formula of higher complexity than the first. In this case, one is forced to remove the falsified knowledge and all its consequences of greater level from the current state. The termination of this add-and-remove process is not at all evident, yet it may be proved by well-established techniques: we shall be able to see that a realizer of an atomic formula is an update procedure in the sense of Avigad [8], and that will be enough for witness extraction. As in [4], one can see Interactive realizability as new use of Friedman's translation and modified realizability, that allows to extract update procedures from classical proofs without transforming them in quantifier-free form, as in the epsilon substitution method or Herbrand analysis.

Plan of the Paper. In section §2 we introduce the term calculus $\mathcal{T}_{\text{Class}}$ in which Interactive, learning-based realizers are written, namely an extension of Gödel's system T plus Skolem function symbols for a denumerable collection of Skolem functions. In section §3, we extend Interactive realizability, as described in [1, 4], to $\mathsf{HA}^{\omega} + \mathsf{EM} + \mathsf{SK}$, an arithmetical system with functional variables which includes first-order classical Peano Arithmetic and Skolem axioms. In section §4 we show how to perform witness extraction with Interactive realizability. Full proofs of all results of this paper may be found in [7].

Acknowledgments. We thank Stefano Berardi for valuable comments and suggestions.

2 The Term Calculus $\mathcal{T}_{\text{Class}}$

In this section we follow the approach of [1, 4] and describe the typed lambda calculi $\mathcal{T}$ and $\mathcal{T}_{\text{Class}}$ in which interactive realizers are written. $\mathcal{T}$ is an extension of Gödel's system T (see Girard [15]) with some syntactic sugar. The basic objects of $\mathcal{T}$ are numerals, booleans, and its basic computational constructs are primitive recursion at all types, if-then-else, pairs, as in Gödel's T. $\mathcal{T}$ also includes as basic objects finite partial functions over $\mathbb{N}$ and simple primitive recursive operations over them. $\mathcal{T}_{\text{Class}}$ is obtained from $\mathcal{T}$ by adding on top of it a collection of Skolem function symbols $\Phi_0, \Phi_1, \Phi_2 \ldots$, of type $\mathtt{N} \to \mathtt{N}$, one for each arithmetical formula. The symbols are inert from the computational point of view and realizers are always computed with respect to some approximation of the Skolem maps represented by $\Phi_0, \Phi_1, \ldots$.

2.1 Updates

In order to define $\mathcal{T}$, we start by introducing the concept of "update", which is nothing but a finite function over $\mathbb{N}$ (i.e. a map over $\mathbb{N}$ with finite domain). Realizers of atomic formulas will return these finite functions, or "updates", as new pieces of information that they have learned about the Skolem function $\Phi_0, \Phi_1, \ldots$. Skolem functions, in turn, are used as "oracles" during computations in the system $\mathcal{T}_{\text{Class}}$. Updates are new associations input-output that are intended to correct, and in this sense, to *update*, wrong oracle values used in a computation.

▶ **Definition 1** (Updates and Consistent Union)**.** We define:

1. An update set U, shortly an *update*, is a finite set of triples of natural numbers representing a finite function from $\mathbb{N}^2$ to $\mathbb{N}$.
2. Two triples (a, n, m) and (a', n', m') of numbers are *consistent* if $a = a'$ and $n = n'$ implies $m = m'$. Two updates U_1, U_2 are consistent if $U_1 \cup U_2$ is an update.
3. $\mathbb{U}$ is the set of all updates.
4. The *consistent union* $U_1 \mathcal{U} U_2$ of $U_1, U_2 \in \mathbb{U}$ is $U_1 \cup U_2$ minus all triples of U_2 which are inconsistent with some triple of U_1.

The consistent union $U_1 \mathcal{U} U_2$ is an non-commutative operation: whenever a triple of U_1 and a triple of U_2 are inconsistent, we arbitrarily keep the triple of U_1 and we reject the triple of U_2, therefore for some U_1, U_2 we have $U_1 \mathcal{U} U_2 \neq U_2 \mathcal{U} U_1$. The operator $\mathcal{U}$ represents a way of selecting a consistent subset of $U_1 \cup U_2$, such that $U_1 \mathcal{U} U_2 = \emptyset \implies U_1 = U_2 = \emptyset$. Any operator $\mathcal{U}$ with that property would produce an alternative Realizability Semantics.

2.2 The System $\mathcal{T}$

$\mathcal{T}$ is formally described in figure 1. Terms of the form $\mathsf{if}_A\, t_1\, t_2\, t_3$ will be written in the more legible form $\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3$. A *numeral* is a term of the form $\mathsf{S}(\ldots\mathsf{S}(0)\ldots)$. For every update $U \in \mathbb{U}$, there is in $\mathcal{T}$ a constant $\overline{U} : \mathtt{U}$, where $\mathtt{U}$ is a new base type representing $\mathbb{U}$. We write $\varnothing$ for $\overline{\emptyset}$. In $\mathcal{T}$, there are four operations involving updates (see figure 1):

1. The first operation is denoted by the constant $\mathsf{min} : \mathtt{U} \to \mathtt{N}$. min takes as argument an update constant $\overline{U}$; it returns the minimum numeral a such that $(a, n, m) \in U$ for some $n, m \in \mathtt{N}$, if any exists; it returns 0 otherwise.
2. The second operation is denoted by the constant $\mathsf{get} : \mathtt{U} \to \mathtt{N}^3 \to \mathtt{N}$. get takes as arguments an update constant $\overline{U}$ and three numerals a, n, l; it returns m if $(a, n, m) \in U$ for some $m \in \mathtt{N}$ (i.e. if (a, n) belongs to the domain of the partial function U); it returns l otherwise.
3. The third operation is denoted by the constant $\mathsf{mkupd} : \mathtt{N}^3 \to \mathtt{U}$. mkupd takes as arguments three numerals a, n, m and transforms them into (the constant coding in $\mathcal{T}$) the update $\{(a, n, m)\}$.
4. The fourth operation is denoted by the constant $\Cup : \mathtt{U}^2 \to \mathtt{U}$. $\Cup$ takes as arguments two update constants and returns the update constant denoting their consistent union.

We observe that the constants $\mathsf{min}, \mathsf{get}, \mathsf{mkupd}$ are just syntactic sugar and may be avoided by coding finite partial functions into natural numbers. System $\mathcal{T}$ may thus be coded in Gödel's T. As proved in [1, 4], $\mathcal{T}$ is strongly normalizing, has the uniqueness-of-normal-form property and the following normal form theorem also holds.

▶ **Lemma 2** (Normal Form Property for $\mathcal{T} + C + \mathcal{R}$)**.** *Assume that $\mathcal{R}$ is a functional set of reduction rules for C. Assume A is either an atomic type or a product type. Then any closed normal term $t \in \mathcal{T}$ of type A is: a numeral $n : \mathtt{N}$, or a boolean $\mathtt{True}, \mathtt{False} : \mathtt{Bool}$, or an update constant $\overline{U} : \mathtt{U}$, or a constant of type A, or a pair $\langle u, v\rangle : B \times C$.*

Types

$$\sigma, \tau ::= \mathtt{N} \mid \mathtt{Bool} \mid \mathtt{U} \mid \sigma \to \tau \mid \sigma \times \tau$$

Constants

$$c ::= \mathsf{R}_\tau \mid \mathsf{if}_\tau \mid 0 \mid \mathsf{S} \mid \mathtt{True} \mid \mathtt{False} \mid \mathsf{min} \mid \mathsf{get} \mid \mathsf{mkupd} \mid \Cup \mid \overline{U}\ (\forall U \in \mathbb{U})$$

Terms

$$t, u ::= c \mid x^\tau \mid tu \mid \lambda x^\tau u \mid \langle t, u \rangle \mid \pi_0 u \mid \pi_1 u$$

Typing Rules for Variables and Constants

$$x^\tau : \tau \mid 0 : \mathtt{N} \mid \mathsf{S} : \mathtt{N} \to \mathtt{N} \mid \mathtt{True} : \mathtt{Bool} \mid \mathtt{False} : \mathtt{Bool} \mid \overline{U} : \mathtt{U}\ (\text{for every } U \in \mathbb{U}) \mid \Cup : \mathtt{U} \to \mathtt{U} \to \mathtt{U}$$
$$\mid \mathsf{min} : \mathtt{U} \to \mathtt{N} \mid \mathsf{get} : \mathtt{U} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N} \mid \mathsf{mkupd} : \mathtt{N} \to \mathtt{N} \to \mathtt{N} \to \mathtt{U}$$
$$\mid \mathsf{if}_\tau : \mathtt{Bool} \to \tau \to \tau \to \tau \mid \mathsf{R}_\tau : \tau \to (\mathtt{N} \to (\tau \to \tau)) \to \mathtt{N} \to \tau$$

Typing Rules for Composed Terms

$$\frac{t : \sigma \to \tau \qquad u : \sigma}{tu : \tau} \qquad \frac{u : \tau}{\lambda x^\sigma u : \sigma \to \tau} \qquad \frac{u : \sigma \qquad t : \tau}{\langle u, t \rangle : \sigma \times \tau} \qquad \frac{u : \tau_0 \times \tau_1}{\pi_i u : \tau_i}\ i \in \{0, 1\}$$

Reduction Rules All the usual reduction rules for simply typed lambda calculus (see Girard [15]) plus the rules for recursion, if-then-else and projections

$$\mathsf{R}_\tau uv0 \mapsto u \quad \mathsf{R}_\tau uv\mathsf{S}(t) \mapsto vt(\mathsf{R}_\tau uvt) \quad \mathsf{if}_\tau\, \mathtt{True}\, u\, v \mapsto u \quad \mathsf{if}_\tau\, \mathtt{False}\, u\, v \mapsto v \quad \pi_i \langle u_0, u_1 \rangle \mapsto u_i, i = 0, 1$$

plus the following ones, assuming a, n, m, l be numerals:

$$\mathsf{min}\, \overline{U} \mapsto \begin{cases} a & \text{if } \exists m, n.\ (a, n, m) \in U \wedge \forall (b, i, j) \in U.\ a \leq b \\ 0 & \text{otherwise} \end{cases} \qquad \overline{U_1} \Cup \overline{U_2} \mapsto \overline{U_1 \mathcal{U} U_2}$$

$$\mathsf{get}\, \overline{U}\, a\, n\, l \mapsto \begin{cases} m & \text{if } \exists m.\ (a, n, m) \in U \\ l & \text{otherwise} \end{cases} \qquad \mathsf{mkupd}\, a\, n\, m \mapsto \overline{\{(a, n, m)\}}$$

Figure 1 The extension $\mathcal{T}$ of Gödel's system T.

2.3 The System $\mathcal{T}_{\mathsf{Class}}$

We now define a classical extension of $\mathcal{T}$, that we call $\mathcal{T}_{\mathrm{Class}}$, with a Skolem function symbol for each arithmetical formula. The elements of $\mathcal{T}_{\mathrm{Class}}$ will represent (non-computable) realizers.

▶ **Definition 3** (The System $\mathcal{T}_{\mathsf{Class}}$)**.** Define $\mathcal{T}_{\mathrm{Class}} = \mathcal{T} + \mathcal{SC}$, where $\mathcal{SC}$ is a countable set of Skolem function constants, each one of type $\mathtt{N} \to \mathtt{N}$. We assume to have an enumeration $\Phi_0, \Phi_1, \Phi_2, \ldots$ of all the constants in $\mathcal{SC}$ (while generic elements of $\mathcal{SC}$ will be denoted with letters $\Phi, \Psi, \sigma, \tau, \ldots$).

Every $\Phi \in \mathcal{SC}$ represents a *Skolem function* for some arithmetical formula $\exists y^{\mathtt{N}} A(x, y)$, taking as argument a number x and returning some y such that $A(x, y)$ is true if any exists, and an arbitrary value otherwise. In general, there is no set of computable reduction rules for the constants in $\mathcal{SC}$, and therefore no set of computable reduction rules for $\mathcal{T}_{\mathrm{Class}}$. Each (in general, non-computable) term $t \in \mathcal{T}_{\mathrm{Class}}$ is associated to a set $\{t[s] \mid s \in \mathcal{T}, s : \mathtt{N}^2 \to \mathtt{N}\} \subseteq \mathcal{T}$ of computable terms we call its "approximations", one for each term $s : \mathtt{N}^2 \to \mathtt{N}$ of $\mathcal{T}$, which is thought as a sequence $s_0, s_1, s_2, \ldots$ of computable approximations of the oracles $\Phi_0, \Phi_1, \Phi_2, \ldots$ (with s_i we denote $s(i)$).

▶ **Definition 4** (Approximation at state s)**.** We define:

1. A *state* is a closed term of type $\mathtt{N}^2 \to \mathtt{N}$ of $\mathcal{T}$. If i is a numeral, with s_i we denote $s(i)$.
2. Assume $t \in \mathcal{T}_{\mathrm{Class}}$ and s is a state. The "approximation of t at state s" is the term $t[s]$ of $\mathcal{T}$ obtained from t by replacing each constant Φ_i with s_i.

3 An Interactive Learning-Based Notion of Realizability for $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$

In this section we introduce a learning-based notion of realizability for $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$, Heyting Arithmetic in all finite types (see e.g. Troelstra [20]) plus Excluded Middle and Skolem axiom schemes for all arithmetical formulas. Then we prove our main Theorem, the Adequacy Theorem: *"if a closed formula is provable in* $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$*, then it is realizable"*.

We first define the formal system $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$. We represent atomic predicates of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ with closed terms of $\mathcal{T}_{\text{Class}}$ of type $\mathtt{Bool}$. Terms of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ are elements of $\mathcal{T}_{\text{Class}}$ and thus may include the function symbols in $\mathcal{SC}$. We assume having in Gödel's T some terms $\Rightarrow_{\mathtt{Bool}}\colon \mathtt{Bool} \to \mathtt{Bool} \to \mathtt{Bool}$, $\neg_{\mathtt{Bool}} : \mathtt{Bool} \to \mathtt{Bool}$, $\vee_{\mathtt{Bool}} : \mathtt{Bool} \to \mathtt{Bool} \to \mathtt{Bool} \ldots$, implementing boolean connectives. As usual, we shall use infix notation: for example, we write $t_1 \Rightarrow_{\mathtt{Bool}} t_2$ in place of $\Rightarrow_{\mathtt{Bool}} t_1 t_2$ and similarly for the other connectives.

3.1 Language of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$

We now define the language of the arithmetical theory $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$.

▶ **Definition 5** (Language of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$)**.** The language $\mathcal{L}_{\text{Class}}$ of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ is defined as follows.

1. The terms of $\mathcal{L}_{\text{Class}}$ are all $t \in \mathcal{T}_{\text{Class}}$.
2. The atomic formulas of $\mathcal{L}_{\text{Class}}$ are all $Q \in \mathcal{T}_{\text{Class}}$ such that $Q : \mathtt{Bool}$.
3. The formulas of $\mathcal{L}_{\text{Class}}$ are built from atomic formulas of $\mathcal{L}_{\text{Class}}$ by the connectives $\vee$, $\wedge$, $\to$, $\diagdown$, $\forall$, $\exists$ as usual, with quantifiers possibly ranging over variables $x^\tau, y^\tau, z^\tau, \ldots$ of arbitrary finite type τ of $\mathcal{T}_{\text{Class}}$.
4. A formula of $\mathcal{L}_{\text{Class}}$ is said *arithmetical* if it does not contain constants in $\mathcal{SC}$ and all its quantifiers range over the type $\mathtt{N}$, i.e. it has one of the following forms: $\forall x^{\mathtt{N}} A$, $\exists x^{\mathtt{N}} A$, $A \vee B$, $A \wedge B$, $A \to B$, $A \diagdown B$, P, with A, B arithmetical and P atomic formula of $\mathcal{T}$.

We denote with $\bot$ the atomic formula $\mathtt{False}$ and with $\neg A$ the formula $A \to \bot$. The connective $\diagdown$ is the dual of implication as in bi-intuitionistic logic and means "A and the opposite of B". If F is a formula of $\mathcal{L}_{\text{Class}}$ in the free variables $x_1^{\tau_1}, \ldots, x_n^{\tau_n}$ and $t_1 : \tau_1, \ldots, t_n : \tau_n$ are terms of $\mathcal{L}_{\text{Class}}$, with $F(t_1, \ldots, t_n)$ we shall denote the formula $F[t_1/x_1, \ldots, t_n/x_n]$. Sequences of variable $x_1^{\mathtt{N}}, \ldots, x_k^{\mathtt{N}}$ will be written as $\vec{x}$. We denote with $\langle \vec{x} \rangle$ a term of $\mathcal{T}$ in the free numeric variables $\vec{x}$ representing a injection of $\mathtt{N}^k$ into $\mathtt{N}$. Moreover, for every sequence of numerals $\vec{n} = n_1, \ldots, n_k$, we define $\langle \vec{n} \rangle := \langle \vec{x} \rangle [\vec{n}/\vec{x}]$ and assume that the function $\vec{n} \mapsto \langle \vec{n} \rangle$ is a bijection.

The *Excluded Middle axiom scheme* EM is defined as the set of all formulas of the form:

$$\forall \vec{x}^{\mathtt{N}}.\ A(\vec{x}) \vee \neg A(\vec{x})$$

where A is an arithmetical formula.

The *Skolem axiom scheme* SK contains for each arithmetical formula $A(\vec{x}, y)$ an axiom:

$$\forall \vec{x}^{\mathtt{N}}.\ \exists y^{\mathtt{N}} A(\vec{x}, y) \to A(\vec{x}, \Phi\langle \vec{x} \rangle)$$

with $\Phi \in \mathcal{SC}$. We assume that for every $\Phi \in \mathcal{SC}$ there is in SK one and only one formula in which Φ occurs. Such unique formula A is said to be the *formula associated to* Φ and Φ will be sometimes written as Φ_A. If s is a state and $\Phi_i = \Phi_A$, with s_A we denote s_i and with $\mathsf{mkupd}\, A\, u\, t$ we denote $\mathsf{mkupd}\, i\, u\, t$. With $\mathsf{lev}(\Phi)$ we denote the number measuring the logical complexity of the formula A associated to Φ, i.e. the number of quantifiers occurring in A.

If s, s' are two states and n a numeral, we write $s \equiv s'\,\mathsf{lev}(n)$ if for every A of complexity strictly less than n, one has $s_A(m) = s'_A(m)$ for all numerals m.

For each formula F of $\mathcal{L}_{\text{Class}}$, its involutive negation $F^\perp$ is defined by induction on F. First, we say that an atomic formula P is positive if it is of the form $\neg_{\mathtt{Bool}} \dots \neg_{\mathtt{Bool}} Q$, Q is not of the form $\neg_{\mathtt{Bool}} R$, and the number of $\neg_{\mathtt{Bool}}$ in front of Q is even. Then we define:

$$
\begin{array}{ll}
(\neg_{\mathtt{Bool}} P)^\perp = P \text{ (if } P \text{ positive)} & P^\perp = \neg_{\mathtt{Bool}} P \text{ (if } P \text{ positive)} \\
(A \wedge B)^\perp = A^\perp \vee B^\perp & (A \vee B)^\perp = A^\perp \wedge B^\perp \\
(A \to B)^\perp = A \setminus B & (A \setminus B)^\perp = A \to B \\
(\forall x^\tau A)^\perp = \exists x^\tau A^\perp & (\exists x^\tau A)^\perp = \forall x^\tau A^\perp
\end{array}
$$

As usual, one has $(F^\perp)^\perp = F$.

3.2 Truth Value of a Formula in a State

The axioms of the system $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ give a great computational power to the system $\mathcal{T}_{\text{Class}}$: one can "compute" by a term χ_F of $\mathcal{T}_{\text{Class}}$ the truth value of any arithmetical formula F. When one effectively evaluates χ_F in a particular state s, we say that one computes *the truth value of a formula F in a state s*.

▶ **Definition 6** (Truth Value of a Formula F in a State s)**.** For every arithmetical formula $F(\vec{x})$ of $\mathcal{L}_{\text{Class}}$ we define, by induction on F, a term $\chi_F : \mathtt{Bool}$ of system $\mathcal{T}_{\text{Class}}$, with the same free variables of F:

$$
\begin{array}{lll}
\chi_P = P,\ P \text{ atomic} & & \\
\chi_{A \vee B} = \chi_A \vee_{\mathtt{Bool}} \chi_B & \chi_{\forall y^{\mathbb{N}} A} = \chi_A[\Phi_{A^\perp}\langle \vec{x} \rangle / y] & \chi_{A \setminus B} = \chi_A \wedge_{\mathtt{Bool}} \chi_{B^\perp} \\
\chi_{A \wedge B} = \chi_A \wedge_{\mathtt{Bool}} \chi_B & \chi_{\exists y^{\mathbb{N}} A} = \chi_A[\Phi_A \langle \vec{x} \rangle / y] & \chi_{A \to B} = \chi_A \Rightarrow_{\mathtt{Bool}} \chi_B
\end{array}
$$

We define $F^s := \chi_F[s]$ and call it *the truth value of F in the state s*.

Intuitively, if $F(\vec{n})$ is a closed formula, our intended interpretation is:

1. $\chi_F(\vec{n})$ is a term of $\mathcal{T}_{\text{Class}}$ *denoting*, in any standard model of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$, the truth value of $F(\vec{n})$.
2. $F^s(\vec{n})$ is a term of $\mathcal{T}$ *computing what would be* the truth value of $F(\vec{n})$ in some standard model of $\mathsf{HA}^\omega + \mathsf{EM}$ *under the (possibly false) assumption* that the interpretation $\Phi_i \mapsto s_i$ satisfies the axioms of SK.

We remark that thus $F^s(\vec{n})$ is only a *conditional* truth value: if $F^s(\vec{n})$ is not the correct truth value of $F(\vec{n})$ – it may well happen – then the interpretation $\Phi_i \mapsto s_i$ does not satisfy the axioms of SK. This subtle point is what makes possible learning in Interactive realizability: whenever a contradiction follows, realizers are able to effectively find counterexamples to the assertion that the interpretation $\Phi_i \mapsto s_i$ satisfies the axioms of SK. We also observe that this way of computing the truth of a formula comes from the epsilon substitution method (see Avigad [8], Mints et al. [18]).

The notion of truth in a state behaves as expected with respect to involutive negation.

▶ **Proposition 7** (Truth in a State and Truth)**.** *For every arithmetical formula $F(\vec{x})$, state s and sequence of numerals $\vec{n}$, $F^s(\vec{n}) = \mathtt{False} \iff (F^\perp)^s(\vec{n}) = \mathtt{True}$*

We now prove a fundamental fact: the truth of a formula F in a state s is determined by the approximations that s gives to the Skolem functions of strictly lower level than the logical complexity of F.

▶ **Proposition 8.** *Let $F(\vec{x})$ be any arithmetical formula of logical complexity m, $\vec{n}$ be a sequence of numerals and s, s' be states such that $s \equiv s' \,\mathsf{lev}(m)$. Then $F^s(\vec{n}) = F^{s'}(\vec{n})$.*

Every state s is considered as an *approximation* of the Skolem functions denoted by the constants of $\mathcal{SC}$: for each formula A, s_A may be a correct approximation of Φ_A on some arguments, but wrong on other ones. More precisely, if $\Phi_i = \Phi_A$, we are going to consider the set of $(i, \langle\vec{n}\rangle)$ such that $A^s(\vec{n}, s_A\langle\vec{n}\rangle) = \mathtt{True}$ as the real "domain" of s, representing the set of arguments at which s_A is surely a correct approximation of Φ_A, in the sense that s_A returns an appropriate witness (but observe that the truth of A is approximated in s).

▶ **Definition 9** (Sound Updates, Domains)**.** We define:

1. Given an update U and a state s, we define $\mathsf{dom}_s(U)$ as the set of pairs of numerals $(i, \langle\vec{n}\rangle)$ such that $A(\vec{x}, y)$ is the formula associated to Φ_i, $(i, \langle\vec{n}\rangle, m) \in U$ and $A^s(\vec{n}, m) = \mathtt{True}$. U is said to be *sound in the state s* if $(i, \langle\vec{n}\rangle, m) \in U$ implies $(i, \langle\vec{n}\rangle) \in \mathsf{dom}_s(U)$.
2. Similarly, if s is a state, we denote with $\mathsf{dom}(s)$ the set of pairs of numerals $(i, \langle\vec{n}\rangle)$ such that $A(\vec{x}, y)$ is the formula associated to Φ_i, $s_i\langle\vec{n}\rangle = m$ and $A^s(\vec{n}, m) = \mathtt{True}$.

From now onwards, for every pair of terms t_1, t_2 of system $\mathcal{T}$, we shall write $t_1 = t_2$ if they are the same term modulo the equality rules corresponding to the reduction rules of system $\mathcal{T}$ (equivalently, if they have the same normal form).

3.3 Interactive Realizability

For every formula A of $\mathcal{L}_{\text{Class}}$, we now define what type $|A|$ a realizer of A must have.

▶ **Definition 10** (Types for realizers)**.** For each formula A of $\mathcal{L}_{\text{Class}}$ we define a type $|A|$ of $\mathcal{T}_{\text{Class}}$ by induction on A:

$$|P| = \mathtt{U}, \text{ if } P \text{ is atomic}$$

$$|A \wedge B| = |A| \times |B| \qquad |\exists x^\tau A| = \tau \times |A| \qquad |A \setminus B| = |A| \times |B^\perp|$$

$$|A \vee B| = \mathtt{Bool} \times (|A| \times |B|) \qquad |\forall x^\tau A| = \tau \to |A| \qquad |A \to B| = |A| \to |B|$$

Let now $\mathsf{p}_0 := \pi_0 : \sigma_0 \times (\sigma_1 \times \sigma_2) \to \sigma_0$, $\mathsf{p}_1 := \pi_0\pi_1 : \sigma_0 \times (\sigma_1 \times \sigma_2) \to \sigma_1$ and $\mathsf{p}_2 := \pi_1\pi_1 : \sigma_0 \times (\sigma_1 \times \sigma_2) \to \sigma_2$ be the three canonical projections from $\sigma_0 \times (\sigma_1 \times \sigma_2)$. We define the realizability relation $t \Vdash F$, where $t \in \mathcal{T}_{\text{Class}}$, $F \in \mathcal{L}_{\text{Class}}$ and $t : |F|$.

▶ **Definition 11** (Interactive Realizability)**.** Assume s is a state, t is a closed term of $\mathcal{T}_{\text{Class}}$, $F \in \mathcal{L}_{\text{Class}}$ is a closed formula, and $t : |F|$. We define first the relation $t \Vdash_s F$ by induction and by cases according to the form of F:

1. $t \Vdash_s Q$ for some atomic Q if and only if $\overline{U} = t[s]$ implies:
   - U is sound in s and $\mathsf{dom}_s(U) \cap \mathsf{dom}(s) = \emptyset$
   - $\overline{U} = \varnothing$ implies $Q[s] = \mathtt{True}$
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 $\mathsf{p}_0 t[s] = \mathtt{True}$ and $\mathsf{p}_1 t \Vdash_s A$, or $\mathsf{p}_0 t[s] = \mathtt{False}$ and $\mathsf{p}_2 t \Vdash_s B$
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. $t \Vdash_s A \setminus B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B^\perp$
6. $t \Vdash_s \forall x^\tau A$ if and only if for all closed terms $u : \tau$ of $\mathcal{T}$, $tu \Vdash_s A[u/x]$
7. $t \Vdash_s \exists x^\tau A$ if and only for some closed term $u : \tau$ of $\mathcal{T}$, $\pi_0 t[s] = u$ and $\pi_1 t \Vdash_s A[u/x]$

We define $t \Vdash F$ if and only if for all states s of $\mathcal{T}$, $t \Vdash_s F$.

The ideas behind the definition of $\Vdash_s$ in the case of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ are those we already explained in [1, 4, 6]. A realizer is a term t of $\mathcal{T}_{\text{Class}}$, possibly containing some non-computable Skolem function of $\mathcal{SC}$; if such a function was computable, t would be an intuitionistic realizer. Since in general t is not computable, we calculate its approximation $t[s]$ at state s. t is an intelligent, self-correcting program, representing a proof/construction depending on the state s. The realizer *interacts* with the environment, which may provide a counter-proof, a counterexample invalidating the current construction of the realizer. But the realizer is always able to turn such a negative outcome into a positive information, which consists in some new piece of knowledge learned about some Skolem function Φ_i.

There are two concepts that are useful to understand the interaction of a realizer with the environment: a realizer receives as input *tests* and produces as output *predictions*.

- *Predictions.*
  - A realizer t of $A \vee B$ uses s to predict which one between A and B is realizable: if $\pi_0 t[s] = \texttt{True}$ then A is realizable, and if $\pi_0 t[s] = \texttt{False}$ then B is realizable.
  - A realizer u of $\exists x^\tau A$ uses s to compute $\pi_0 u[s] = w$ and to predict that w is a witness for $\exists x^\tau A$ (i.e. that $A[w/x]$ is realizable).
- *Tests.*
  - A realizer of a universal formula $\forall x^\tau A$ takes an object w as a challenge coming from the environment to provide a construction of $A[w/x]$, whose correctness will be tested at the end of computation.
  - A realizer of $A \to B$ takes a realizer of A as a challenge coming from the environment to provide a construction of B, whose correctness will be tested at the end of the computation.
  - A realizer of $A \wedge B$ may be challenged to construct A as well as B, and again the correctness of the construction will be tested at the end of computation.
  - A realizer of an atomic formula Q comes after a series of predictions and challenges that have been provided to test the construction of a complex formula; the realizer performs a final test and computes the formula Q in the state s as an experiment. Since predictions of realizers need not be always correct, it is possible that a realized atomic formula is actually false; we may have $t \Vdash_s Q$ and $Q[s] = \texttt{False}$ in $\mathcal{T}$. If Q, though predicted to be true, is instead false, then a counterexample has been encountered; this means that the approximation s of the Skolem constants in $\mathcal{SC}$ is still inadequate. In this case, $t[s] \neq \varnothing$ by definition of $t \Vdash_s Q$. That is to say: if the construction of a realizer is wrong in a particular state, the realizer must learn from its mistakes. The point is that after every learning, the actual state can be improved with the information in $\overline{U} = t[s]$, since $(i, \langle \vec{n} \rangle) \in U$ and A is associated to Φ_i imply $A^s(\vec{n}, m) = \texttt{True}$ and $A^s(\vec{n}, s_i\langle \vec{n} \rangle) = \texttt{False}$.

The next proposition tells that realizability at state s respects the notion of equality of $\mathcal{T}_{\text{Class}}$ terms, when the latter is relativized to state s. That is, if two terms are equal at the state s, then they realize the same formulas in the state s.

▶ **Proposition 12** (Saturation). *If $t_1[s] = t_2[s]$ and $u_1[s] = u_2[s]$, then $t_1 \Vdash_s B[u_1/x]$ if and only if $t_2 \Vdash_s B[u_2/x]$.*

3.4 Realizability of Classical Axioms

We are now going to show how to realize EM and SK. We first need to realize the ex-falso-quodlibet axiom, for which a dummy term is enough.

▶ **Proposition 13** (Realizer of the Ex-Falso-Quodlibet Axiom). *For every formula $F(\vec{x})$ of $\mathcal{L}_{Class}$, there exists a closed term $\bot_F$ of $\mathcal{T}$ such that $\bot_F \Vdash \bot \to F(\vec{u})$, for every sequence of closed terms $\vec{u}$ of $\mathcal{T}_{Class}$. In particular, $\bot_F$ can be defined by induction on F as follows:*

$$\begin{aligned}
\bot_P &:= \lambda x^{\mathtt{U}}.\, x & \bot_{A\to B} &:= \lambda x^{\mathtt{U}} \lambda y^{|A|}.\, \bot_B x \\
\bot_{A\wedge B} &:= \lambda x^{\mathtt{U}}.\langle \bot_A x, \bot_B x\rangle & \bot_{\exists x^\tau A} &:= \lambda x^{\mathtt{U}} \langle 0^\tau, \bot_A x\rangle \\
\bot_{A\vee B} &:= \lambda x^{\mathtt{U}}.\langle \mathtt{False}, \bot_A x, \bot_B x\rangle & \bot_{\forall x^\tau A} &:= \lambda x^{\mathtt{U}} \lambda y^\tau.\bot_A x \\
\bot_{A \setminus B} &:= \lambda x^{\mathtt{U}} \langle \bot_A x, \bot_{B^\bot} x\rangle
\end{aligned}$$

In Interactive realizability, as we shall see many times, realizers continually interact with each other and game theory is a very useful tool to describe they behaviour (see [5]). Interestingly, for realizing Skolem axioms and some instances of EM we are lead to consider once again Tarski games. In these kind of games, there are two players and an arithmetical formula on the board; the first player – usually called Eloise – tries to show that it is true, while the second player – usually called Abelard – tries to show that the formula is false. Thus, Eloise wins when true atomic formulas are on the board while Abelard wins with false ones. In the case of formulas of the shape $A \vee B, \exists x^{\mathtt{N}} A$, Eloise moves: in the first case by choosing a side of the disjunction and in the second case by choosing a numeral as a witness for the existential quantifier. In the case of formulas of the shape $A \wedge B, \forall x^{\mathtt{N}} A$, Abelard moves: in the first case by choosing a side of the conjunction and in the second case by choosing a numeral as a counterexample to the universal quantifier. In the case of formulas of the shape $A \to B$, Abelard gives a winning strategy for A to Eloise and they play the game for B. An Eloise strategy for A is represented by a term $\mathscr{E}$ of type $|A|$, while an Abelard strategy for A is represented by a term $\mathscr{A}$ of type $|A^\bot|$. Thus one may define the Tarski game between strategies through a game operator $\star$ (which indeed resembles the operator $\star$ of symmetric lambda calculus [9]) that puts Eloise against Abelard. The result of the game is thus $\mathscr{E} \star \mathscr{A}$ and it is a term of type $|\bot| = \mathtt{U}$. If $\mathscr{E}$ and $\mathscr{A}$ happens to be interactive realizers, they represent self-correcting constructions. $\mathscr{E}$ and $\mathscr{A}$ challenge the construction of each other, but who looses the interaction is always able to partially repair its construction, i.e. to learn some information about some Skolem functions, which he puts in some update. That is to say, $\mathscr{E} \star \mathscr{A}$ realizes $\bot$. Details follow.

▶ **Proposition 14** (A Tarski Game Between Strategies). *Let F be an arithmetical formula and $\mathscr{E} : |F|$, $\mathscr{A} : |F^\bot|$ two terms of $\mathcal{T}_{Class}$. Define by induction and according to the shape of F a term $\mathscr{E} \star \mathscr{A} : |\bot|$ as follows:*

$$\begin{aligned}
&(F = P, P \text{ atomic})\ \mathscr{E} \star \mathscr{A} := \mathscr{E} \Cup \mathscr{A} \\
&(F = A \to B)\ \mathscr{E} \star \mathscr{A} := \mathscr{E}(\pi_0 \mathscr{A}) \star \pi_1 \mathscr{A} && (F = A \setminus B)\ \mathscr{E} \star \mathscr{A} := \mathscr{A}(\pi_0 \mathscr{E}) \star \pi_1 \mathscr{E} \\
&(F = \exists y^{\mathtt{N}} A)\ \mathscr{E} \star \mathscr{A} := \pi_1 \mathscr{E} \star \mathscr{A}(\pi_0 \mathscr{E}) && (F = \forall y^{\mathtt{N}} A)\ \mathscr{E} \star \mathscr{A} := \mathscr{E}(\pi_0 \mathscr{A}) \star \pi_1 \mathscr{A}
\end{aligned}$$

$$\begin{aligned}
&(F = A \wedge B)\ \mathscr{E} \star \mathscr{A} := \mathsf{if}\ \mathsf{p}_0 \mathscr{A}\ \mathsf{then}\ \pi_0 \mathscr{E} \star \mathsf{p}_1 \mathscr{A}\ \mathsf{else}\ \pi_1 \mathscr{E} \star \mathsf{p}_2 \mathscr{A} \\
&(F = A \vee B)\ \mathscr{E} \star \mathscr{A} := \mathsf{if}\ \mathsf{p}_0 \mathscr{E}\ \mathsf{then}\ \mathsf{p}_1 \mathscr{E} \star \pi_0 \mathscr{A}\ \mathsf{else}\ \mathsf{p}_2 \mathscr{E} \star \pi_1 \mathscr{A}
\end{aligned}$$

Then $\mathscr{E} \Vdash_s F \wedge \mathscr{A} \Vdash_s F^\bot \implies \mathscr{E} \star \mathscr{A} \Vdash_s \bot$.

We now establish two important links between the concept of truth in a state and the concept of realizability in the same state.

The first result is that if a formula is true in a state s, then it is realizable in s. Intuitively, $F^s = \mathtt{True}$ means that the state s is both: i) powerful enough to compute witnesses for all the subformulas of F and their negations which are supposed to be true if F is true; ii) sharp

enough to not provide counterexamples invalidating some of those witnesses. Thus F can be realized in the state s by a realizer $\mathfrak{T}_F$ which uses s to return the mentioned witnesses and "waits" for possible counterexamples.

One could expect the converse to hold as well, namely that if F is realizable in s, then F is true in s. This is not actually true, but "almost". Indeed, the second result is that if F is realizable in s and $F^s = \texttt{False}$, one has disastrous consequences: $\bot$ is realizable in s. In fact, one can define a term $\mathfrak{F}_F$ transforming any realizer $\mathscr{E}$ of F in the state s in a realizer of $\bot$ in the same s, whenever $F^s = \texttt{False}$. Indeed, if $F^s = \texttt{False}$, then $(F^\bot)^s = \texttt{True}$ by proposition 7; therefore $\mathscr{A} := \mathfrak{T}_{F^\bot}$ realizes $F^\bot$ in s. The state s is thus used to build a counterexample $\mathscr{A}$ to F, which can be put against the realizer $\mathscr{E}$ of F in the term $\mathscr{E} \star \mathscr{A}$. This latter term realizes $\bot$ by proposition 14. Intuitively, if s is a strong enough approximation, $\mathscr{A}$ wins and thus provides a counterexample to the fact that $\mathscr{E}$ is a construction of F; since $\mathscr{E}$, as realizer, is a self-correcting program, it is able to extend the state s with new information: this information realizes $\bot$ in s. If instead s is not a good approximation, $\mathscr{E}$ wins and the capabilities of $\mathscr{A}$ are used to improve the state s with an update realizing $\bot$. Formally:

▶ **Proposition 15** (Truth and Realizability in a State). *Let $F(\vec{x})$ be an arithmetical formula. There exist two terms $\mathfrak{T}_F(\vec{x})$ and $\mathfrak{F}_F(\vec{x})$ of $\mathcal{T}_{Class}$ such that for all numerals $\vec{n}$ and state s*

$$F^s(\vec{n}) = \texttt{True} \implies \mathfrak{T}_F(\vec{n}) \Vdash_s F(\vec{n})$$
$$F^s(\vec{n}) = \texttt{False} \implies \mathfrak{F}_F(\vec{n}) \Vdash_s \neg F(\vec{n})$$

In particular, $\mathfrak{T}_F$ and $\mathfrak{F}_F$ can be constructed by induction on F as follows:

$$\mathfrak{T}_P := \varnothing,\ P \text{ atomic} \qquad \mathfrak{T}_{A\to B} := \lambda z^{|A|}.\,\mathsf{if}\ \chi_A\ \mathsf{then}\ \mathfrak{T}_B\ \mathsf{else}\ \bot_B(\mathfrak{F}_A z)$$
$$\mathfrak{T}_{A\wedge B} := \langle \mathfrak{T}_A, \mathfrak{T}_B\rangle \qquad \mathfrak{T}_{\forall y^{\mathbb{N}} A} := \lambda y^{\mathbb{N}}.\,\mathsf{if}\ \chi_A\ \mathsf{then}\ \mathfrak{T}_A\ \mathsf{else}\ \bot_A \mathsf{mkupd}\, A^\bot\,\langle\vec{x}\rangle\, y$$
$$\mathfrak{T}_{A\vee B} := \langle \chi_A, \mathfrak{T}_A, \mathfrak{T}_B\rangle \qquad \mathfrak{T}_{\exists y^{\mathbb{N}} A} := \langle \Phi_A\langle\vec{x}\rangle, \mathfrak{T}_A[\Phi_A\langle\vec{x}\rangle/y]\rangle$$
$$\mathfrak{T}_{A\setminus B} := \langle \mathfrak{T}_A, \mathfrak{T}_{B^\bot}\rangle \qquad \mathfrak{F}_F := \lambda x^{|F|}.\, x \star \mathfrak{T}_{F^\bot}$$

The most remarkable feature of our realizability semantics is the existence of a realizer for any instance of EM, even if our language contains the positive symbols $\vee, \exists$ which have to be realized according to the Kreisel-style clauses of our definition of realizability.

▶ **Proposition 16** (Realizer E_A of EM). *Let $A(\vec{x})$ be any arithmetical formula. Define*

$$\mathsf{E}_A := \lambda\vec{x}^{\mathbb{N}}.\,\langle \chi_A, \mathfrak{T}_A, \mathfrak{F}_A\rangle$$

Then $\mathsf{E}_A \Vdash \forall\vec{x}^{\mathbb{N}}.\, A(\vec{x}) \vee \neg A(\vec{x})$.

We observe that the excluded middle can very well be defined as $\forall\vec{x}^{\mathbb{N}}.\, A(\vec{x}) \vee A^\bot(\vec{x})$. In the case of Σ^0_n-formulas, one would get a simplified realizer of the form $\lambda\vec{x}^{\mathbb{N}}.\,\langle \chi_A, \mathfrak{T}_A, \mathfrak{T}_{A^\bot}\rangle$, which does not use the game operator $\star$ contained in $\mathfrak{F}_A$. In the case of Σ^0_1-formulas, one recovers as a special case exactly the realizer of [1], where Interactive realizability was first defined for $\mathsf{HA} + \mathsf{EM}_1 + \mathsf{SK}_1$. We also observe that the realizer E_A, when evaluated in any state, uses two instructions: the *read from the state* operation for satisfying the constructive clauses of realizability and the *update of the state* operation for dealing with counterexamples to universal quantifiers. The operation of updating the state is used only to interpret classical steps of proofs. We now show how to realize the Skolem axiom scheme SK.

▶ **Proposition 17** (Realizer S_A of SK). *Let $A(\vec{x}, y)$ be any arithmetical formula and Φ the Skolem function constant associated to A. Define*

$$\mathsf{S}_A := \lambda\vec{x}^{\mathbb{N}}\lambda z^{|\exists y^{\mathbb{N}} A|}.\,\mathsf{if}\ \chi_{\exists y^{\mathbb{N}} A}\ \mathsf{then}\ \mathfrak{T}_A[\Phi\langle\vec{x}\rangle/y]\ \mathsf{else}\ \bot_{A(\vec{x},\Phi\langle\vec{x}\rangle)}(\mathfrak{F}_{\exists y^{\mathbb{N}} A} z)$$

Then $\mathsf{S}_A \Vdash \forall\vec{x}^{\mathbb{N}}.\, \exists y^{\mathbb{N}} A(\vec{x}, y) \to A(\vec{x}, \Phi\langle\vec{x}\rangle)$.

3.5 Curry-Howard Correspondence for $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$

In figure 2, we define a standard natural deduction system for $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ (see [19], for example) together with a term assignment in the spirit of Curry-Howard correspondence for classical logic. We replace purely universal axioms (i.e., Π^0_1-axioms) with Post rules, which

---

Contexts With Γ we denote contexts of the form $x_1 : A_1, \dots, x_n : A_n$, with $x_1, \dots, x_n$ proof variables and $A_1, \dots, A_n$ formulas of $\mathcal{L}_{\text{Class}}$.

Axioms $\Gamma, x : A \vdash x^{|A|} : A$

Conjunction $\dfrac{\Gamma \vdash u : A \quad \Gamma \vdash t : B}{\Gamma \vdash \langle u, t\rangle : A \wedge B}$ $\qquad \dfrac{\Gamma \vdash u : A \wedge B}{\Gamma \vdash \pi_0 u : A}$ $\qquad \dfrac{\Gamma \vdash u : A \wedge B}{\Gamma \vdash \pi_1 u : B}$

Subtraction $\dfrac{\Gamma \vdash u : A \quad \Gamma \vdash t : B^\perp}{\Gamma \vdash \langle u, t\rangle : A \setminus B}$ $\qquad \dfrac{\Gamma \vdash u : A \setminus B}{\Gamma \vdash \pi_0 u : A}$ $\qquad \dfrac{\Gamma \vdash u : A \setminus B}{\Gamma \vdash \pi_1 u : B^\perp}$

Implication $\dfrac{\Gamma \vdash u : A \to B \quad \Gamma \vdash t : A}{\Gamma \vdash ut : B}$ $\qquad \dfrac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x^{|A|} u : A \to B}$

Disjunction Intro. $\dfrac{\Gamma \vdash u : A}{\Gamma \vdash \langle \mathtt{True}, u, d^{|B|}\rangle : A \vee B}$ $\qquad \dfrac{\Gamma \vdash u : A}{\Gamma \vdash \langle \mathtt{False}, d^{|A|}, u\rangle : A \vee B}$

Disjunction Elim. $\dfrac{\Gamma \vdash u : A \vee B \quad \Gamma, x : A \vdash w_1 : C \quad \Gamma, x : B \vdash w_2 : C}{\Gamma \vdash \mathsf{if}\ \mathsf{p}_0 u\ \mathsf{then}\ (\lambda x^{|A|} w_1)(\mathsf{p}_1 u)\ \mathsf{else}\ (\lambda x^{|B|} w_2)(\mathsf{p}_2 u) : C}$

Universal Quantification $\dfrac{\Gamma \vdash u : \forall \alpha^\tau A}{\Gamma \vdash ut : A[t/\alpha^\tau]}$ $\qquad \dfrac{\Gamma \vdash u : A}{\Gamma \vdash \lambda \alpha^\tau u : \forall \alpha^\tau A}$

where t is a term of $\mathcal{L}_{\text{Class}}$ and α^{N} does not occur free in any formula B occurring in Γ.

Existential Quantification $\dfrac{\Gamma \vdash u : A[t/\alpha^\tau]}{\Gamma \vdash \langle t, u\rangle : \exists \alpha^\tau . A}$ $\qquad \dfrac{\Gamma \vdash u : \exists \alpha^\tau . A \quad \Gamma, x : A \vdash t : C}{\Gamma \vdash (\lambda \alpha^\tau \lambda x^{|A|}\ t)(\pi_0 u)(\pi_1 u) : C}$

where α^τ is not free in C nor in any formula B occurring in Γ.

Induction $\dfrac{\Gamma \vdash u : A(0) \quad \Gamma \vdash v : \forall \alpha^{\mathsf{N}} . A(\alpha) \to A(\mathsf{S}(\alpha))}{\Gamma \vdash \lambda \alpha^{\mathsf{N}} \mathsf{R} u v \alpha : \forall \alpha^{\mathsf{N}} A}$

Post Rules $\dfrac{\Gamma \vdash u_1 : A_1\ \Gamma \vdash u_2 : A_2\ \cdots\ \Gamma \vdash u_n : A_n}{\Gamma \vdash u_1 \Cup u_2 \Cup \cdots \Cup u_n : A}$

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, for a Peano axiom or for a classical propositional tautology or for booleans.

Post Rules with no Premises $\dfrac{}{\Gamma \vdash \varnothing : A}$

where A is an atomic formula of $\mathcal{L}_{\text{Class}}$ and an axiom of equality or a classical propositional tautology.

EM $\dfrac{}{\Gamma \vdash \mathsf{E}_A : \forall \vec{x}^{\mathsf{N}}.\ A(\vec{x}) \vee \neg A(\vec{x})}$

SK $\dfrac{}{\Gamma \vdash \mathsf{S}_A : \forall \vec{x}^{\mathsf{N}}.\ \exists y^{\mathsf{N}} A(\vec{x}, y) \to A(\vec{x}, \Phi\langle \vec{x}\rangle)}$

Figure 2 Term Assignement Rules for $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$.

are inferences of the form

$$\frac{\Gamma \vdash A_1\ \Gamma \vdash A_2\ \cdots\ \Gamma \vdash A_n}{\Gamma \vdash A}$$

where $A_1, \dots, A_n, A$ are atomic formulas of $\mathcal{L}_{\text{Class}}$ such that for every substitution $\sigma = [t_1/x_1, \dots, t_k/x_k]$ of closed terms $t_1, \dots, t_k$ of $\mathcal{T}$ and state s, $A_1\sigma[s] = \ldots = A_n\sigma[s] = \mathtt{True}$ implies $A\sigma[s] = \mathtt{True}$. Let now $\mathsf{eq} : \mathtt{N}^2 \to \mathtt{Bool}$ be a term of Gödel's system T representing equality between natural numbers. Among the Post rules, we have the Peano axioms and axioms of equality

$$\frac{\Gamma \vdash \mathsf{eq}\ \mathsf{S}(x)\,\mathsf{S}(y)}{\Gamma \vdash \mathsf{eq}\ x\,y} \qquad \frac{\Gamma \vdash \mathsf{eq}\ 0\,\mathsf{S}(x)}{\Gamma \vdash \bot} \qquad \frac{}{\Gamma \vdash \mathsf{eq}\ x\,x} \qquad \frac{\Gamma \vdash \mathsf{eq}\,x\,y \quad \Gamma \vdash \mathsf{eq}\,y\,z}{\Gamma \vdash \mathsf{eq}\ x\,z} \qquad \frac{\Gamma \vdash A(x)\ \Gamma \vdash \mathsf{eq}\,x\,y}{\Gamma \vdash A(y)}$$

and for every A_1, A_2 such that $A_1 = A_2$ is an equation of Gödel's system T (equivalently, A_1, A_2 have the same normal form in T), we have the rule

$$\frac{\Gamma \vdash A_1}{\Gamma \vdash A_2}$$

We also add a Post rule for every classical propositional tautology $A_1 \to \ldots \to A_n \to A$, where for $i = 1, \ldots, n$, A_i, A are atomic formulas obtained as combination of other atomic formulas by the Gödel's system T boolean connectives. Finally, we have a rule of case reasoning for booleans. For any atomic formula P and any formula $A[P]$ we have:

$$\frac{\Gamma \vdash A[\mathtt{True}] \quad \Gamma \vdash A[\mathtt{False}]}{\Gamma \vdash A[P]}$$

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

It can now be proved that every theorem of $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$ is realizable (see [7]).

▶ **Theorem 18.** *If A is a closed formula such that $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK} \vdash t : A$, then $t \Vdash A$.*

4 Witness Extraction with Interactive Realizability

In this section, we turn our attention to the witness extraction problem for Π^0_2-formulas. Given a realizer $t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} P x y$, where P is an atomic recursive predicate, one is asked to extract from t a non-trivial program taking as input a numeral n and yielding as output a witness for the formula $\exists y^{\mathtt{N}} P n y$ (that is, a numeral m such that $Pnm = \mathtt{True}$). In the case of Interactive realizability, the problem of computing that witness can be reduced to finding a "zero" for a suitable term u of type $\mathtt{U}$, that is a state s such that $u[s] = \varnothing$. Indeed, given any numeral n and state s, the following implications hold:

$$t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} P x y \implies t \Vdash_s \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} P x y \implies tn \Vdash_s \exists y^{\mathtt{N}} P n y$$

$$\implies (\pi_0(tn)[s] = m \wedge \pi_1(tn) \Vdash_s Pnm) \implies (\pi_1(tn)[s] = \varnothing \implies Pnm = \mathtt{True})$$

Therefore, if s is a zero of $\pi_1(tn)$, then $\pi_0(tn)$ is equal in the state s to some witness m of the formula $\exists y^{\mathtt{N}} P n y$. Intuitively, a zero for $\pi_1(tn)$ represents a sufficient amount of information to compute the required witness. Indeed, a zero for $\pi_1(tn)$ always exists, because $\pi_1(tn)$ represents an update procedure (see [2, 8] for investigations and explanations of the concept).

▶ **Definition 19** (Avigad's Finite Update Procedures). A $k+1$-*ary typed update procedure* $k \in \mathbb{N}$ is a term $\mathbf{U} : (\mathtt{N} \to \mathtt{N})^k \to \mathtt{U}$ of $\mathcal{T}$ such that the following holds:

1. for all sequences $f = f_0, \ldots, f_k$ of closed type-$\mathtt{N} \to \mathtt{N}$ terms of $\mathcal{T}$

$$\mathbf{U}f \neq \varnothing \implies \mathbf{U}f = \{(i, n, m)\} \wedge 0 \leq i \leq k$$

2. for all sequences $f = f_0, \ldots, f_k$ and $g = g_0, \ldots, g_k$ of closed type-$\mathtt{N} \to \mathtt{N}$ terms of $\mathcal{T}$ and for all $0 \leq i \leq k$, if
   - $\mathbf{U}f = \{(i, n, m)\}$
   - for all $j < i$, $f_j = g_j$
   - $g_i(n) = m$

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

If $\mathbf{U}$ is a $k+1$-ary update procedure, a *zero* for $\mathbf{U}$ is a sequence $f = f_0, \ldots, f_k$ of closed type-$\mathtt{N} \to \mathtt{N}$ terms of $\mathcal{T}$ such that $\mathbf{U}f = \varnothing$.

Condition (2) describes *learning by levels*. If $\mathbf{U}$ is a k-ary update procedure and f is a sequence of terms approximating the Skolem functions $\Phi_0, \ldots, \Phi_k$ (and we assume $i \leq j$, implies $\mathsf{lev}(\Phi_i) \leq \mathsf{lev}(\Phi_j)$), there are two possibilities: either f is a fine approximation and then $\mathbf{U}f = \emptyset$; or f is not and then $\mathbf{U}f = \{(i, n, m)\}$, for some numerals n, m: $\mathbf{U}$ says the term f_i should be updated as to output m on input n. Moreover, if $\mathbf{U}f = (i, n, m)$, one in a sense has *learned* at level i that $\Phi_i(n) = m$ on grounds of the values of f_j, for $j < i$. Condition (2) indicates that the information $\Phi_i(n) = m$ should be preserved, unless some information in the lower levels changes.

For technical convenience we now add to $\mathcal{T}$ (and thus to system $\mathcal{T}_{\text{Class}}$), a constant $\mathsf{ch} : \mathtt{U} \to \mathtt{U}$ which chooses exactly one element from every non-empty update and maps the empty update to itself. Therefore, we assume to have in $\mathcal{T}$ conversion rules such that for every non-empty update U

$$\mathsf{ch}\, \varnothing \mapsto \varnothing \qquad \mathsf{ch}\, \overline{U} \mapsto \{(i, n, m)\}, \text{ for some } (i, n, m) \in U$$

For simplicity, we are going to consider only *proof-like terms* of $\mathcal{T}_{\text{Class}}$: a term t is said to be proof-like if: i) every occurrence in t of the constant mkupd is of the form mkupd_i, where i is some numeral, and Φ_i occurs in t; ii) no update constant different from $\varnothing$ occurs in t. Indeed, that is the syntactic form of every interactive realizer extracted from some proof in $\mathsf{HA}^\omega + \mathsf{EM} + \mathsf{SK}$.

▶ **Proposition 20** (The Update Procedure Associated to an Atomic Realizer)**.** *Let Q be an atomic formula of $\mathcal{L}_{Class}$ and suppose $t \Vdash Q$, with t proof-like. Let without loss of generality $\Phi_0, \ldots, \Phi_k$ the list of all Skolem function constants of t ordered by levels: if $i \leq j$, then $\mathsf{lev}(\Phi_i) \leq \mathsf{lev}(\Phi_j)$. Define*

$$\mathbf{U} := \lambda f_0^{\mathtt{N}\to\mathtt{N}} \ldots \lambda f_k^{\mathtt{N}\to\mathtt{N}}.\, \mathsf{ch}(t[f_0/\Phi_0 \ldots f_k/\Phi_k])$$

Then $\mathbf{U}$ is an update procedure.

From now on, the term $\mathbf{U}$ of proposition 20 will be called the $(k+1)$-*ary update procedure associated to* t. There is a standard way to compute a zero for any update procedure and thus for the correspondent atomic realizer. In order to do that, if $f = f_0, \ldots, f_k$ is a sequence of terms of type $\mathtt{N} \to \mathtt{N}$ and $\overline{U}$ is an update constant, we define a term $f \oplus \overline{U}$ which changes the values of f_i according to the triples $(i, n, m) \in U$, where $i = \mathsf{min}\, \overline{U}$, leaves f_j for $j < i$ unchanged and changes every f_j, with $j > i$, to be equal to the constant function $\lambda x^{\mathtt{N}} 0$.

▶ **Definition 21** (Updates of Functions)**.** For each numeral i, we define a term $\oplus_i : (\mathtt{N} \to \mathtt{N}) \to \mathtt{U} \to (\mathtt{N} \to \mathtt{N})$ as follows:

$$\oplus_i := \lambda f^{\mathtt{N}\to\mathtt{N}} \lambda u^{\mathtt{U}} \lambda x^{\mathtt{N}} \text{ if } \mathsf{min}\, u > i \text{ then } fx \text{ else if } \mathsf{min}\, u = i \text{ then } (\mathsf{get}\, u\, i\, x\, fx) \text{ else } 0$$

We shall write $t_1 \oplus_i t_2$ in place of $\oplus_i t_1 t_2$. If $f = f_0, \ldots, f_k$ is a sequence of terms of type $\mathtt{N} \to \mathtt{N}$ and $u : \mathtt{U}$, we define $f \oplus u := f_0 \oplus_0 u, \ldots, f_k \oplus_k u$.

▶ **Theorem 22** (Zero Theorem)**.** *Let Q be an atomic formula of $\mathcal{L}_{Class}$, suppose t is a proof-like term such that $t \Vdash Q$ and let $\mathbf{U}$ be the $(k+1)$-ary update procedure associated to t. Let s be any state. Define, by induction on n, a sequence $\{r_n\}_{n\in\mathbb{N}}$ of $k+1$-ary sequences of type-$\mathtt{N} \to \mathtt{N}$ terms as follows:*

$$r_0 := s_0, \ldots, s_k$$
$$r_{n+1} := r_n \oplus (\mathbf{U} r_n) \text{ (see definition 21)}$$

Then, there exists a n such that $t[(r_n)_0/\Phi_0 \ldots (r_n)_k/\Phi_k] = \varnothing$.

Using the results of [3, 6], we are even able to extract a program belonging to system $\mathcal{T}$.

▶ **Theorem 23** (Program Extraction via Interactive Realizability)**.** *Let t be a term of $\mathcal{T}_{Class}$ and suppose that $t \Vdash \forall x^{\mathtt{N}} \exists y^{\tau} Pxy$, with $P : \mathtt{N} \to \tau \to \mathtt{Bool}$ closed term of system $\mathcal{T}$. Then:*

1. *From t one can effectively define a recursive function f such that for every numeral n, $f(n) : \tau$ is a term of system $\mathcal{T}$ such that $Pn(f(n)) = \mathtt{True}$.*
2. *f can be represented in system $\mathcal{T}$.*

Remark. We observe that our algorithm for witness extraction is not the last word on the topic, for it is not particularly optimized for real-world execution. However, thanks to our realizability interpretation, we have now achieved a sharp understanding and control of the learning process which is implicit in every computational interpretation of classical logic. This is crucial: the inefficiency of the algorithms extracted from classical proofs is usually due to their inability to backtrack without forgetting important information that they have acquired during the computation. It is already evident that dramatically more efficient algorithm are possible, by managing in a more sophisticated way the update of the states. For example, multiple updates of the states can be allowed at one time and in the proof of the Zero theorem one does not need to "set to zero" every approximation corresponding to a Skolem function of higher level of the first level which is corrected by an update. For reasons of space, we leave these optimizations to future work.

References

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

2 F. Aschieri, *Transfinite Update Procedures for Predicative Systems of Analysis*, Proceedings of Computer Science Logic, 2011.

3 F. Aschieri, *A Constructive Analysis of Learning in Peano Arithmetic*, Annals of Pure and Applied Logic, 2011, doi:10.1016/j.apal.2011.12.004.

4 F. Aschieri, S. Berardi, *A New Use of Friedman's Translation: Interactive Realizability*, Festschrift of Helmut Schwichtenberg, Ontos-Verlag Series in Mathematical Logic, to appear.

5 F. Aschieri, *Learning Based Realizability for HA + EM1 and 1-Backtracking Games: Soundness and Completeness*, Annals of Pure and Applied Logic, to appear.

6 F. Aschieri, *Interactive Realizability for Second-Order Heyting Arithmetic with* EM_1 *and* SK_1, Technical Report, http://hal.inria.fr/hal-00657054.

7 F. Aschieri, *Interactive Realizability for Classical Peano Arithmetic with Skolem Axioms*, Technical Report, http://hal.inria.fr/hal-00685360.

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

9 F. Barbanera, S. Berardi, *A Symmetric Lambda-Calculus for Classical Program Extraction*, Information and Computation, 1996.

10 S. Berardi and U. de' Liguoro, *Interactive Realizers. A New Approach to Program Extraction from Nonconstructive Proofs*, ACM Transactions on Computational Logic, 2012.

11 T. Coquand, *A Semantic of Evidence for Classical Arithmetic*, Journal of Symbolic Logic, 1995.

12 H. Friedman, *Classically and Intuitionistically Provable Recursive Functions*, Lecture Notes in Mathematics, 1978, Volume 669/1978, 21-27.

13 K. Gödel, *Uber eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes*, Dialectica 12, pp. 280-287 (1958).

14 G. Gentzen, *Die Widerspruchsfreiheit der reinen Zahlentheorie*. Mathematische Annalen,1935.

15 J.-Y. Girard, *Proofs and Types*, Cambridge University Press (1989).

16 G. Kreisel, *On Weak Completeness of Intuitionistic Predicate Logic*, Journal of Symbolic Logic, vol. 27, 1962.

17 J-L. Krivine, *Typed lambda-calculus in classical Zermelo-Fraenkel set theory*, Archive for Mathematical Logic, 40(3), 2001.

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

19 M. H. Sorensen, P. Urzyczyn, *Lectures on the Curry-Howard isomorphism*, Studies in Logic and the Foundations of Mathematics, vol. 149, Elsevier, 2006.

20 A. Troelstra, D. van Dalen, *Constructivism in Mathematics, vol. I*, North-Holland, 1988.