Math. Struct. in Comp. Science (2014), vol. 24, iss. 6, e240601, 34 pages. © Cambridge University Press 2013
doi:10.1017/S0960129513000455

Interactive Realizability for second-order Heyting arithmetic with EM1 and SK1

FEDERICO ASCHIERI

Laboratoire de l'Informatique du Parallélisme (UMR 5668), équipe Plume,
École Normale Supérieure de Lyon – Université de Lyon,
46 Allée d'Italie, 69007 Lyon, France
Email: federico.aschieri@ens-lyon.fr

Received 3 March 2012; revised 23 May 2013

We introduce a realizability semantics based on interactive learning for full second-order Heyting arithmetic with excluded middle and Skolem axioms over Σ^0_1-formulas. Realizers are written in a classical version of Girard's System F and can be viewed as programs that learn by interacting with the environment. We show that the realizers of any Π^0_2-formula represent terminating learning processes whose outcomes are numerical witnesses for the existential quantifier of the formula.

1. Introduction

In recent years, several computational interpretations of classical logic have been put forward. Under a first classification, they can be grouped into two large categories: *direct* and *indirect* interpretations. Among the indirect interpretations, there are the negative translations followed either by Dialectica interpretations (Gödel 1958; Spector 1962) or intuitionistic realizability interpretations combined with Friedman's translation (Friedman 1978). Among the direct interpretations, there are classical realizabilities, Coquand game semantics (Coquand 1995), cut-elimination and normalisation of classical proofs (under Curry–Howard correspondence or not), and the epsilon substitution method (Mints *et al.* 1996) – the Kreisel no-counterexample interpretation (Kreisel 1951) can be obtained as an easy corollary of the others.

Such variety is surprising, and at first sight these interpretations may appear completely different. However, it is becoming evident that some unifying concepts exist, and possibly the most general and powerful of them is the concept of *learning*. That is, the computational content of classical proofs can be described in terms of learning programs, which acquire new knowledge about non-computable functions by an intelligent process of making hypotheses and testing them in search of counterexamples. As soon as we adopt this conceptual perspective, all the computational interpretations become clearly related and appear as technical variations on the same theme: learning.

On the one hand, indirect interpretations yield, as a result of negative translation, programs using continuations. The deep reason continuations are used is that they are natural tools for implementing backtracking, that is, the mechanism by which learning programs make guesses, learn about their mistakes and correct them thanks to the newly acquired knowledge. Berardi *et al.* (1998) used a realizability interpretation of the

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

negative translation of the axiom of countable choice with continuations to build finite approximations of choice functions. From the computational point of view, the Dialectica interpretation and the Friedman translation provide ways to capture counterexamples to the hypotheses made by learning programs.

On the other hand, direct interpretations exploit the fact that classical principles have a remarkably immediate computational content when considered as learning devices. In the case of Krivine classical realizability (Krivine 2009), the excluded middle $A \vee \neg A$ is interpreted as a program that assumes $\neg A$ as a working hypothesis: if at some point of the computation the program encounters evidence for A (that is, a realizer of A), then it backtracks, erases everything that depended on the hypothesis $\neg A$ and acquires a realizer of A as new knowledge. Coquand game semantics interprets the excluded middle in basically the same spirit, but in a more semantically meaningful way. Avigad's presentation of the epsilon substitution method (Avigad 2002) instead exploits the learning content of Skolem axioms, which are formulas of the form

$$\forall x \forall y . A(x, y) \implies A(x, f(x)).$$

The function f is interpreted as an approximation of some choice function mapping x to a witness (if any exists) for the formula $\exists y A(x, y)$. Whenever an instance

$$A(n, m) \implies A(n, f(n))$$

of a Skolem axiom is false (according to some current approximation of the truth values of $A(n, m)$ and $A(n, f(n))$) the function f can be corrected so as to output m on input n.

1.1. *Realizability based on interactive learning*

Given this strong evidence that learning is the *key* to understanding the computational content of classical proofs, an important goal is to formulate a classical realizability semantics explicitly based on learning. Such a semantics should first describe the nature of the knowledge that programs coming from classical proofs acquire during computations, and then say how this knowledge evolves during computations. As a consequence of this approach, it should be possible to develop a much finer understanding and control of the backtracking mechanism that interprets classical proofs: in other words, more efficient programs.

A significant step towards this goal was taken in Aschieri and Berardi (2010) and Aschieri and Berardi (2012), which introduced a learning-based classical realizability for first-order Heyting arithmetic HA (also in its finite-type version HA^ω) with the excluded middle EM_1 on Σ^0_1-formulas and Skolem axioms SK_1 over quantifier-free formulas. This is a realizability based on states describing the current knowledge of realizers. The reason such a fragment was isolated and studied is that we only need to use *monotonic* learning to interpret it, where monotonic learning means that learning programs can only increase their knowledge, and, once acquired, that knowledge is correct forever. This is

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

the simplest type of learning. It has special properties[†] and is worth studying in its own right. In game semantics, it is represented as *simple backtracking* – see Coquand (1991, page 90) and Aschieri (2013). In more general settings, learning is more complex. For example, in the case of full first-order Peano arithmetic, learning is finitely *nested*, that is knowledge is stratified and the correctness of what is learned at any level depends on what has been learned at the previous levels – see Avigad (2002) for an ordinal analysis of this kind of learning, and Aschieri (2011a) and Aschieri (2012) for a type-theoretic analysis. In the case of predicative fragments of analysis, learning is transfinitely nested (Aschieri 2011b).

The crucial contribution of Interactive Realizability is that it decomposes the extraction of programs from classical proofs into two conceptual steps. The idea is that we first extract an ideal program, obtained with free use of oracles and Skolem functions: this program is very natural to write, obeys the laws of intuitionistic Heyting semantics (see, for example, Troelstra and van Dalen (1988)) and is easy to understand. Then, classical principles suggest how we can very efficiently approximate the oracles and Skolem functions used in the computation. The result is a model of intelligent programs, which can correct themselves and learn from the mistakes they make when trying to achieve some goal defined by the usual Heyting intuitionistic reading of logical sentences. With hindsight, our interpretation can be viewed as a modern version of the epsilon substitution method, refined and rebuilt around the Curry–Howard correspondence for classical logic.

In the current paper, we extend Interactive Realizability for $\mathsf{HA} + \mathsf{EM}_1 + \mathsf{SK}_1$ to second-order Heyting arithmetic HAS plus EM_1 and SK_1. The arithmetic $\mathsf{HA} + \mathsf{EM}_1 + \mathsf{SK}_1$ is realized by adding an oracle for the halting problem to Gödel's system T, and by computationally interpreting EM_1 as a device that effectively learns oracle values during calculations. The resulting notion of realizability is just Kreisel modified realizability (Kreisel 1959) extended with learning (Aschieri and Berardi 2012). It is thus natural to realize the theory $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$ by adding to Girard's system F an oracle for the halting problem and then try again to interpret the excluded middle as a learning device: this is indeed the approach we will follow in the current paper. Again, the resulting notion of realizability will be a natural extension of intuitionistic realizability for HAS (for which we basically follow the formulation of Oliva and Streicher (2008), but in a Church style) with learning.

We also introduce a technique for witness extraction from proofs of Π^0_2-formulas. That is, given a realizer

$$t \Vdash \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} P(x, y),$$

with P an atomic predicate, we extract a non-trivial program taking as input any number n and yielding as output a number m such that $P(n, m)$ holds. In particular, the program

[†] In the field of proof mining, we are indeed interested in exploiting every special property of the theory being studied. If a constructive interpretation generalises directly to a stronger theory, there is a good chance that some constructive information about the theorems of the former theory will be lost.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

uses t as a state-extending operator, and we shall prove that it is enough to extend the state a finite number of times to get enough information to compute a witness.

1.2. *Plan of the paper*

In Section 2, we introduce the term calculus in which realizers will be written, namely an extension of Girard's system F plus a constant symbol for a Skolem function Φ.

In Section 3, we introduce our notion of realizability based on interactive learning for $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$.

In Section 4, we present our technique for witness extraction.

2. The term calculus $\mathcal{F}_{\text{Class}}$

In this section we introduce the typed lambda calculi we shall use to define Interactive Realizability: system $\mathcal{F}$ and $\mathcal{F}_{\text{Class}}$. System $\mathcal{F}$ is a completely standard extension of Girard's system F (Girard 1989) with some syntactic sugar covering numerals, booleans, primitive recursion at all types, if-then-else, pairs, finite partial functions over $\mathbb{N}$ and simple primitive recursive operations over them. Equivalently, $\mathcal{F}$ may be seen as an extension of Gödel's system T (as in the exposition of Girard (1989)) with polymorphism. System $\mathcal{F}_{\text{Class}}$ is obtained from $\mathcal{F}$ by adding on top of it a Skolem function symbol $\Phi : \mathtt{N} \to (\mathtt{N} \to \mathtt{N})$ of the same Turing degree as an oracle for the halting problem. The symbol is totally inert from the computational point of view, so realizers will always be computed with respect to some approximation of the Skolem function represented by Φ.

2.1. *Updates*

In order to define $\mathcal{F}$, we must first define the concept of update, which is just a finite partial function over $\mathbb{N}$. We use the name 'update' because realizers of atomic formulas will return finite partial functions as new pieces of information that they have learned about the Skolem function Φ. Updates represent new input–output associations that are intended to correct (and in this sense *update*) wrong oracle values used in computations.

Definition 2.1 (updates and consistent union). We define:

(1) An update set U (an *update* for short) is a finite set of triples of natural numbers representing a finite partial 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 $m = m'$.
(3) Two updates U_1, U_2 are consistent if $U_1 \cup U_2$ is an update.
(4) $\mathbb{U}$ is the set of all updates.
(5) 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 that are inconsistent with some triple of U_1.

We think of a triple (a,n,m) contained in an update as the code of a possible witness m for a Σ^1_0-formula $\exists y.\mathsf{P}_a(n,y)$. The fact that every update is a partial *function* allows at most one witness in each update for each formula $\exists y.\mathsf{P}_a(n,y)$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

$U_1 \mathcal{U} U_2$ is a 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 reject the triple of U_2, so for some U_1, U_2, we have $U_1 \mathcal{U} U_2 \neq U_2 \mathcal{U} U_1$. The operation $\mathcal{U}$ is a 'learning strategy', which is a way of selecting a consistent subset of $U_1 \cup U_2$.

It is trivial to show that $\mathcal{U}$ is an associative operation on the set of updates, with neutral element $\varnothing$, with upper bound $U_1 \cup U_2$ and returning a non-empty update whenever $U_1 \cup U_2$ is non-empty.

Lemma 2.2. Assume $i \in \mathbb{N}$ and $U_1, \ldots, U_i \in \mathbb{U}$. Then:

(1) $U_1 \mathcal{U} \ldots \mathcal{U} U_i \subseteq U_1 \cup \ldots \cup U_i$.
(2) $U_1 \mathcal{U} \ldots \mathcal{U} U_i = \varnothing$ implies $U_1 = \ldots = U_i = \varnothing$.

In fact, the whole realizability semantics is a monad (Berardi and de' Liguoro 2008). Berardi and de' Liguoro (2008) proved that a fragment of our realizability semantics is parametric with respect to the definition we choose for $\mathcal{U}$. Any associative operation $\mathcal{U}$ with neutral element $\varnothing$ and satisfying the two properties of Lemma 2.2 defines a different but sound realizability semantics, each of which corresponds to a different 'learning strategy'.

2.2. *System $\mathcal{F}$*

System $\mathcal{F}$ is formally described in Figure 1. A numeral is a term of the form $\mathsf{S}(\mathsf{S}(\ldots 0))$. For notational simplicity, we assume in the following that updates are made of triples of numerals. Terms of the form $\mathsf{if}\ T\, t_1\, t_2\, t_3$ will be written in the more readable form

$$\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3,$$

whenever T can be inferred from the context. For every update $U \in \mathbb{U}$, there is in $\mathcal{F}$ a constant $\overline{U} : \mathtt{U}$, where $\mathtt{U}$ is a new base type representing $\mathbb{U}$. We write $\varnothing$ for $\overline{\varnothing}$. In $\mathcal{F}$, there are four operations involving updates (see Figure 1):

(1) The first operation is denoted by the constant

$$\mathsf{is} : \mathtt{U} \to \mathtt{N}^2 \to \mathtt{Bool}.$$

is takes as arguments an update constant $\overline{U}$ and two numerals a, n. It returns `True` if $(a, n, m) \in U$ for some numeral m (that is, if the pair (a, n) is in the domain of the partial function U), otherwise it returns `False`.

(2) The second operation is denoted by the constant

$$\mathsf{get} : \mathtt{U} \to \mathtt{N}^2 \to \mathtt{N}.$$

get takes as arguments an update constant $\overline{U}$ and two numerals a, n. It returns m if $(a, n, m) \in U$ for some numeral m (that is, if (a, n) belongs to the domain of the partial function U), otherwise it returns 0.

(3) The third operation is denoted by the constant

$$\mathsf{mkupd} : \mathtt{N}^3 \to \mathtt{U}.$$

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Types

$$A, B ::= X \mid \mathtt{N} \mid \mathtt{Bool} \mid \mathtt{U} \mid A \to B \mid A \times B \mid \forall X A$$

Constants

$$c ::= \mathsf{R} \mid \mathsf{if} \mid 0 \mid \mathsf{S} \mid \mathtt{True} \mid \mathtt{False} \mid \mathsf{mkupd} \mid \mathsf{is} \mid \mathsf{get} \mid \ \Cup \ \mid \overline{U} \text{ (for every } U \in \mathbb{U})$$

Terms

$$t, u ::= \ c \mid x^A \mid tu \mid \lambda x^A u \mid tA \mid \Lambda X u \mid \langle t, u \rangle \mid \pi_i u$$

Typing rules for variables and constants

$$x^A : A$$
$$0 : \mathtt{N}$$
$$\mathsf{S} : \mathtt{N} \to \mathtt{N}$$
$$\mathtt{True} : \mathtt{Bool}$$
$$\mathtt{False} : \mathtt{Bool}$$
$$\overline{U} : \mathtt{U} \text{ (for every } U \in \mathbb{U})$$
$$\Cup : \mathtt{U} \to \mathtt{U} \to \mathtt{U}$$
$$\mathsf{is} : \mathtt{U} \to \mathtt{N} \to \mathtt{N} \to \mathtt{Bool}$$
$$\mathsf{get} : \mathtt{U} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N}$$
$$\mathsf{mkupd} : \mathtt{N} \to \mathtt{N} \to \mathtt{N} \to \mathtt{U}$$
$$\mathsf{if} : \forall X.\ \mathtt{Bool} \to X \to X \to X$$
$$\mathsf{R} : \forall X.\ X \to (\mathtt{N} \to (X \to X)) \to \mathtt{N} \to X$$

Typing rules for composed terms

$$\frac{t : A \to B \qquad u : A}{tu : B} \qquad \frac{u : B}{\lambda x^A u : A \to B}$$

$$\frac{u : \forall X A}{uB : A[B/X]} \qquad \frac{u : A}{\Lambda X u : \forall X A} \text{ with } X \text{ non free in the types of the free variables of } u$$

$$\frac{u : A \qquad t : B}{\langle u, t \rangle : A \times B} \qquad \frac{u : A_0 \times A_1}{\pi_i u : A_i}\ i \in \{0, 1\}$$

Reduction rules

We have all the usual reduction rules for system F (Girard 1989) plus the following rules for recursion, if-then-else and projections

$$\mathsf{R}\, T\, u\, v\, 0 \mapsto u \qquad \mathsf{R}\, T\, u\, v\, \mathsf{S}(t) \mapsto v\, t\, (\mathsf{R}\, T\, u\, v\, t)$$
$$\mathsf{if}\, T\, \mathtt{True}\, u\, v \mapsto u \qquad \mathsf{if}\, T\, \mathtt{False}\, u\, v \mapsto v \qquad \pi_i \langle u_0, u_1 \rangle \mapsto u_i, i = 0, 1$$

plus the following rules, where we assume a, n, m are numerals

$$\mathsf{is}\, \overline{U}\, a\, n \mapsto \begin{cases} \mathtt{True} & \text{if for some numeral } m,\ (a, n, m) \in U \\ \mathtt{False} & \text{otherwise} \end{cases}$$

$$\mathsf{get}\, \overline{U}\, a\, n \mapsto \begin{cases} m & \text{if for some numeral } m,\ (a, n, m) \in U \\ 0 & \text{otherwise} \end{cases}$$

$$\overline{U}_1 \Cup \overline{U}_2 \mapsto \overline{U_1 \mathcal{U} U_2}$$

$$\mathsf{mkupd}\, a\, n\, m \mapsto \overline{U}, \text{with } U = \{(a, n, m)\}$$

Fig. 1. System $\mathcal{F}$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

mkupd takes as arguments three numerals a, n, m and transforms them into (the constant coding) the update $\{(a, n, m)\}$.

(4) The fourth operation is denoted by the constant

$$\mathbb{U} : \mathtt{U}^2 \to \mathtt{U}.$$

$\mathbb{U}$ takes as arguments two update constants and returns the update constant denoting their consistent union.

Note that the constants is, get and mkupd are just syntactic sugar, which may be avoided by coding finite partial functions into natural numbers. We assume that there are in system $\mathcal{F}$ some terms

$$\Rightarrow_{\mathtt{Bool}} : \mathtt{Bool} \to \mathtt{Bool} \to \mathtt{Bool}$$
$$\neg_{\mathtt{Bool}} : \mathtt{Bool} \to \mathtt{Bool}$$
$$\vee_{\mathtt{Bool}} : \mathtt{Bool} \to \mathtt{Bool} \to \mathtt{Bool},$$

and so on, which implement boolean connectives. If $t_1, \ldots, t_n, t \in \mathcal{F}$ have type Bool and are made using boolean connectives from free variables all of type Bool, we say that t is a tautological consequence of $t_1, \ldots, t_n$ in $\mathcal{F}$ (or just a tautology if $n = 0$) if all boolean assignments making $t_1, \ldots, t_n$ equal to True in $\mathcal{F}$ also make t equal to True in $\mathcal{F}$. Some terms of Gödel's system T will be used to represent predicates.

Definition 2.3 (predicates of system T).

(1) A binary *predicate* of T is any closed normal term $P : \mathtt{N}^2 \to \mathtt{Bool}$ of Gödel's system T.
(2) We assume $\mathsf{P}_0, \mathsf{P}_1, \mathsf{P}_2, \ldots$ is an arbitrary enumeration of all binary predicates of T.

As usual when working with the polymorphic lambda calculus, we can define sum types $A + B$ and existential types $\exists XA$. We shall need $A + B$ in order to define functionals that return either objects of type A or of type B, without knowing in advance which type it will be. This situation occurs when we need to define a realizer for a disjunction $A \vee B$, which has either to return a realizer of A or a realizer of B. Similarly, we shall need $\exists XA$ in order to define functionals that return a pair of a type B and an object of type $A[B/X]$, without knowing in advance the type B. This situation occurs when we need to define a realizer for a formula $\exists XA$, which has to return a type B and a realizer of $A[B/X]$, for some B.

Definition 2.4 (sum types, existential types).

(1) For all types A, B of system $\mathcal{F}$, we define a type

$$A + B := \forall X.(A \to X) \to (B \to X) \to X$$

where X is a variable not occurring free in A, B.

(2) For every term $u : A$ of system F, we define a term

$$\iota_{0,A,B}(u) := \Lambda X \lambda f^{A \to X} \lambda g^{B \to X} f u$$

of type $A + B$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(3) For every term $u : B$ of system F, we define a term

$$\iota_{1,A,B}(u) := \Lambda X \lambda f^{A\to X} \lambda g^{B\to X} gu$$

of type $A + B$.

(4) For all types A of $\mathcal{F}$, we define a type

$$\exists X A := \forall Y . (\forall X . A \to Y) \to Y$$

where Y is a variable not occurring free in A.

(5) For every type B and term u of type $A[B/X]$, we define a term

$$\langle B, u\rangle := \Lambda Y \lambda x^{\forall X . A\to Y} xBu$$

of type $\exists X A$.

System $\mathcal{F}$ is obtained from system $\mathtt{T}$ by adding polymorphism and new operations on atomic types. The following definition formalises what has been done and is useful for defining arbitrary extensions of $\mathcal{F}$ with arbitrary functions over natural numbers: we shall need such extensions for adjoining non-computable functions to $\mathcal{F}$.

Definition 2.5 (functional set of rules). Let C be any set of constants, each of some type $A_1 \to \ldots \to A_n \to A$, for some $A_1, \ldots, A_n, A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{U}\}$. 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 $\mathcal{F}$, of exactly one rule $ca_1 \ldots a_n \mapsto a$ for some closed normal term $a : A$ of $\mathcal{F}$.

Any extension of $\mathcal{F}$ with constants, and even with non-computable functional sets of rules, is strongly normalising and has the uniqueness-of-normal-form property.

Theorem 2.6. If $\mathcal{R}$ is a functional set of reduction rules for C (Definition 2.5), then $\mathcal{F} + C + \mathcal{R}$ enjoys strong normalisation and weak Church–Rosser (uniqueness of normal forms) for all closed terms of atomic types.

Proof. For strong normalisation, see Berger (2005), or just use standard reducibility arguments, as in Girard (1989). Weak Church–Rosser is also standard. □

The following normal form theorem also holds.

Lemma 2.7 (normal form property for $\mathcal{F} + C + \mathcal{R}$). If $\mathcal{R}$ is a functional set of reduction rules for C and A is either an atomic type or a product type, then any closed normal term $t \in \mathcal{F}$ of type A is:

— a numeral $n : \mathtt{N}$;
— a boolean $\mathtt{True}, \mathtt{False} : \mathtt{Bool}$;
— an update constant $\overline{U} : \mathtt{U}$;
— a constant of type A; or
— a pair $\langle u, v\rangle : B \times C$.

Proof. We use induction over t. We consider cases for t for some sequence $\vec{v}$ of closed types and terms:

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

— $t = (\lambda x.u)\vec{v}$:
Then t has an arrow type if $\vec{v} = \varnothing$, and t is not normal if $\vec{v} \neq \varnothing$.
— $t = (\Lambda X.u)\vec{v}$:
Then t has universal type if $\vec{v} = \varnothing$, and t is not normal if $\vec{v} \neq \varnothing$.
— $t = \langle u, w\rangle\vec{v}$ for some variable x:
Then $\vec{v} = \varnothing$ and we are done.
— $t = x(\vec{v})$
Then t is not closed.
— $t = c\vec{v} : A$ for some constant c:

We have the following subcases:

- If $t = 0$, we are done.
- If $t = \mathsf{S}(u)$, we apply the induction hypothesis to u.
- If $t = \mathtt{True}, \mathtt{False} : \mathtt{Bool}$ or $t = \overline{U} : \mathtt{U}$ or t is a constant of C, we are done.
- The remaining subcases are:

$$t = (\mathsf{R}Us_1s_2n)\vec{t}$$
$$t = (\mathsf{if}\,Uba_1a_2)\vec{t}$$
$$t = \pi_i(z)\vec{w}$$
$$t = \mathsf{is}\,u\,n\,m : \mathtt{N}$$
$$t = \mathsf{get}\,u\,n\,m : \mathtt{N}$$
$$t = \mathbb{U}\,u_1\,u_2 : \mathtt{U}$$
$$t = \mathsf{mkupd}\,n\,m\,l$$
$$t = c\vec{v}, \text{ with } c \in C \text{ and } v_1 : A_1, \ldots, v_k : A_k, \text{ and } A_i \text{ atomic for every } i.$$

The proper subterms $n, m, l : \mathtt{N}$, $b : \mathtt{Bool}$, $z : A \times B$ and $u, u_1, u_2 : \mathtt{U}$, $v_1 : A_1, \ldots, v_k : A_k$ of t have atomic or product type and are closed normal. By the induction hypothesis, they are, respectively, numerals, booleans, pairs and constants. In all cases, t is not normal. □

2.3. *System* $\mathcal{F}_{Class}$

We now define a classical extension of $\mathcal{F}$, which we call $\mathcal{F}_{\text{Class}}$, with a constant symbol $\Phi : \mathtt{N}^2 \to \mathtt{N}$ denoting a non-computable map of the same Turing degree as an oracle for the halting problem. We shall use the elements of $\mathcal{F}_{\text{Class}}$ to represent non-computable realizers.

Definition 2.8 (Systems $\mathcal{F}_{\text{Class}}$ and $\mathcal{T}_{\text{Class}}$). We define

$$\mathcal{F}_{\text{Class}} = \mathcal{F} + \Phi$$
$$\mathcal{T}_{\text{Class}} = \mathsf{T} + \Phi,$$

where $\Phi : \mathtt{N}^2 \to \mathtt{N}$ is a new constant symbol and T is Gödel's system.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

For every numeral a, we have that Φa, which we shall denote by Φ_a, represents a *Skolem function* for the formula $\exists y^{\mathtt{N}}\, \mathsf{P}_a xy$ taking as argument a number x and returning some y such that $\mathsf{P}_a xy$ if any exists, and an arbitrary value otherwise. There is no set of computable reduction rules for the constant Φ, so there is no set of computable reduction rules for $\mathcal{F}_{\text{Class}}$.

Each (in general, non-computable) term $t \in \mathcal{F}_{\text{Class}}$ is associated with a set

$$\{t[s] \mid s \in \mathcal{F}, s : \mathtt{N}^2 \to \mathtt{N}\} \subseteq \mathcal{F}$$

of computable terms, which we call its 'approximations', with one for each term $s : \mathtt{N}^2 \to \mathtt{N}$ of $\mathcal{F}$, and which is thought of as a computable approximation of the oracle Φ.

Definition 2.9 (approximation at state s).

(1) A *state* is a closed term of type $\mathtt{N}^2 \to \mathtt{N}$ of $\mathcal{F}$. We define $\mathtt{S} := \mathtt{N}^2 \to \mathtt{N}$.
(2) Assume $t \in \mathcal{F}_{\text{Class}}$ and s is a state. The 'approximation of t at state s' is the term $t[s]$ of $\mathcal{F}$ obtained from t by replacing each constant Φ with s.

We interpret any $t[s] \in \mathcal{F}$ as a learning process evaluated with respect to the information taken from an approximation s of Φ. Here we consider an approximation of Φ to be an arbitrary term $s : \mathtt{S}$, where s may be correctly in agreement with Φ on some arguments, but wrong on other ones. Consequently, we are going to consider the set of (a, n) such that $\mathsf{P}_a n s_a(n) = \mathtt{True}$ to be the real 'domain' of s (again, we shall sometimes write san to denote $s_a(n)$). We are also going to define a term $\oplus$ that takes as arguments a term $f : \mathtt{N}^2 \to \mathtt{N}$ and an update $\overline{U}$ and changes the values of f according to U. This is one of the fundamental operations of our computational model: realizers will compute updates to correct wrong values of oracle approximations with new good values that they have previously learned and stored in the updates. Finally, using Φ, we are going to define for every numeral a the oracle X_a, which takes as argument a numeral n and returns the truth value of $\exists y^{\mathtt{N}}\, \mathsf{P}_a ny$.

Definition 2.10 (domain, updates of functions, oracle X_a).

(1) If s is a state, we use $\mathsf{dom}(s)$ to denote the set of pairs of numerals (a, n) such that

$$\mathsf{P}_a n s_a(n) = \mathtt{True}.$$

If U is an update, we use $\mathsf{dom}(U)$ to denote the set of pairs of numerals (a, n) such that $(a, n, m) \in U$ and

$$\mathsf{P}_a nm = \mathtt{True}.$$

We say that U is *sound* if for every $(a, n, m) \in U$, we have

$$\mathsf{P}_a nm = \mathtt{True}.$$

(2) We define a term $\oplus : (\mathtt{N}^2 \to \mathtt{N}) \to \mathtt{U} \to (\mathtt{N}^2 \to \mathtt{N})$ as follows:

$$\oplus := \lambda f^{\mathtt{N}^2 \to \mathtt{N}} \lambda u^{\mathtt{U}} \lambda x^{\mathtt{N}} \lambda y^{\mathtt{N}}\ \mathsf{if}\ (\mathsf{is}\, u\, x\, y)\ \mathsf{then}\ (\mathsf{get}\, u\, x\, y)\ \mathsf{else}\ fxy.$$

We will write $t_1 \oplus t_2$ instead of $\oplus t_1 t_2$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(3) For every numeral $a : \mathtt{N}$, we define a term $\mathsf{X}_a : \mathtt{N} \to \mathtt{Bool}$ as follows:

$$\mathsf{X}_a := \lambda x^{\mathtt{N}}\ \mathsf{P}_a x(\Phi_a x).$$

We will now introduce a notion of convergence for families of terms $\{t[s_i]\}_{i\in\mathbb{N}} \subseteq \mathcal{F}$ defined by some $t \in \mathcal{F}_{\text{Class}}$ and indexed over a set $\{s_i\}_{i\in\mathbb{N}}$ of states. Informally, 't convergent' means that the normal form of $t[s_i]$ eventually stops changing when the approximation s_i of Φ gets better and better. If s, r are states, we formalise what it means to say that r is at least as good an approximation as s by defining:

$$s \leqslant r \iff \forall a, n.\ s_a(n) \neq r_a(n) \implies (a,n) \notin \mathsf{dom}(s) \wedge (a,n) \in \mathsf{dom}(r).$$

Intuitively, if $s \leqslant r$, then r can be obtained by correcting some of the values of s that make s a wrong approximation of Φ. We say that a sequence $\{s_i\}_{i\in\mathbb{N}}$ of states is a weakly increasing chain of states (is w.i. for short) if $s_i \leqslant s_{i+1}$ for all $i \in \mathbb{N}$.

Definition 2.11 (convergence). Assume that $\{s_i\}_{i\in\mathbb{N}}$ is a w.i. sequence of states, and $u \in \mathcal{F}_{\text{Class}}$. Then:

(1) u converges in $\{s_i\}_{i\in\mathbb{N}}$ if $\exists i \in \mathbb{N}\ \forall j \geqslant i.\ u[s_j] = u[s_i]$ in $\mathcal{F}$.
(2) u converges if u converges in every w.i. sequence of states.

Note that if u is convergent, we do not require that u be convergent to the *same* value on *all* w.i. chain of oracle approximations. The value attained by u may depend on the information contained in the particular chain from which u gets the knowledge.

Theorem 2.12 (Convergence Theorem). If $t \in \mathcal{F}_{\text{Class}}$ is a closed term of atomic type A ($A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{U}\}$), then t is convergent.

Proof (classical). For every function S mapping pairs of numerals into numerals, we define some (in general, *non*-computable) functional set of reduction rules $\mathcal{R}(S)$ for Φ and for $\mathcal{F}_{\text{Class}}$:

$$\mathcal{R}(S) := \{\Phi_a n \mapsto m \mid S(a,n) = m\}.$$

If s is a state, we define

$$\mathcal{R}(s) := \{\Phi_a n \mapsto m \mid s_a(n) = m, \text{ with } a, n, m \text{ numerals}\}.$$

By Theorem 2.6, for any S and s, we have $\mathcal{F}_{\text{Class}} + \mathcal{R}(S)$ and $\mathcal{F}_{\text{Class}} + \mathcal{R}(s)$ are strongly normalising and weak CR for all closed terms of atomic type.

Claim. If s is a state, the normal form of t in $\mathcal{F}_{\text{Class}} + \mathcal{R}(s)$ is equal to the normal form of $t[s]$ in $\mathcal{F}$.

Proof of the Claim. We use induction over the size of the reduction tree of t.

If t is normal, then, by Lemma 2.7, it is a numeral, a boolean or a constant. Moreover, $t \neq \Phi$, for $\Phi : \mathtt{N}^2 \to \mathtt{N}$. We are now done since $t = t[s]$.

If t is not normal, it will reduce in a one-step reduction to some t'. If we can show that $t[s]$ reduces $t'[s]$ in $\mathcal{F}$, we will get the claim by the induction hypothesis. The only non-self-evident case is when a redex not already in t is contracted in $t[s]$. Such a redex

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

must be of the form $\Phi_a n$, with n numeral, and t' results from t by replacing the redex with m, where m is a numeral and $s_a(n) = m$. But then $t'[s]$ can also be obtained from $t[s]$ by normalising the redex $s_a(n)$ of $t[s]$. □

For the rest of the proof of the theorem, we let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of states and define

$$S_\omega(a,n) := \begin{cases} m & \text{if } \exists i.\ (a,n) \in \mathsf{dom}(s_i) \wedge s_i(a,n) = m \\ s_0(a,n) & \text{otherwise.} \end{cases}$$

S_ω is a well-defined function because $(a,n) \in \mathsf{dom}(s_i)$ and $(a,n) \in \mathsf{dom}(s_j)$ imply

$$s_i(a,n) = s_j(a,n),$$

since $\{s_i\}_{i\in\mathbb{N}}$ is weakly increasing. By strong normalisation, t has a finite reduction in normal form in $\mathcal{F}_{\text{Class}} + \mathcal{R}(S_\omega)$. Therefore, only *finitely many* reduction rules from $\mathcal{R}(S_\omega)$ are used in this reduction. Moreover, if $\Phi_a n \mapsto m$ is any of these rules, then either there exists i such that $(a,n) \in \mathsf{dom}(s_i)$ and $s_i(a,n) = m$, and thus for every $j \geqslant i$, we have

$$s_i(a,n) = s_j(a,n)$$

(for $\{s_i\}_{i\in\mathbb{N}} \in$ w.i.), or it does not, and thus for every j, we have

$$s_j(a,n) = s_0(a,j)$$

(again, $\{s_i\}_{i\in\mathbb{N}} \in$ w.i.). Therefore, all the reduction rules used to obtain the normal form w of t in $\mathcal{F}_{\text{Class}} + \mathcal{R}(S_\omega)$ are already in $\mathcal{R}(s_n)$ for some numeral n, so w is also the normal form of t in $\mathcal{F}_{\text{Class}} + \mathcal{R}(s_n)$, and in $\mathcal{F}_{\text{Class}} + \mathcal{R}(s_m)$ for all $m \geqslant n$. Thus, by the *Claim*, the normal forms in $\mathcal{F}$ of all $t[s_m]$ with $m \geqslant n$ are the same, as we wished to show. □

Remark 2.13. The idea of the proof of Theorem 2.12 corresponds exactly to the intuition that during any computation, the oracle Φ is consulted a finite number of times and hence asked for a finite number of values. When the approximation s_n of Φ is good enough, we can substitute Φ with s_n and obtain the same oracle values, and hence the same results.

3. An interactive learning-based notion of realizability for $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$

In this section we introduce a notion of realizability based on interactive learning for $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$, that is, second-order Heyting arithmetic plus excluded middle on Σ^0_1-formulas

$$\mathsf{EM}_1 := \forall x^{\mathbb{N}}.\ \exists y^{\mathbb{N}} \mathsf{P}_a xy \vee \forall y^{\mathbb{N}} \neg \mathsf{P}_a xy$$

and Skolem axioms

$$\mathsf{SK}_1 := \forall x^{\mathbb{N}} \forall y^{\mathbb{N}}.\ \mathsf{P}_a xy \to \mathsf{P}_a x \Phi_a(x).$$

We will then prove our main theorem, the Adequacy Theorem, which says that *'if a closed arithmetical formula is provable in* $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$*, then it is realizable'*.

We will begin by defining the formal system $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$. We represent atomic predicates of $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$ with (in general, non-computable) closed terms of $\mathcal{T}_{\text{Class}}$ of type `Bool`. Terms of $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$ may include the function symbol Φ, denoting the Skolem function for $\exists y^{\mathbb{N}} \mathsf{P}_a xy$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Remark 3.1. Our realizability can already be formulated for the standard language of arithmetic: we add non-computable functions to the language for greater generality and to interpret Skolem axioms in addition to EM_1. Of course, HAS already proves the comprehension axiom, so $\mathsf{HAS} + \mathsf{EM}_1$ proves the existence of a Skolem function for $\exists y^{\mathbb{N}} \mathsf{P}_a xy$, which is the formula $\exists XA$, where

$$A := \mathsf{fun}(X) \wedge \forall x^{\mathbb{N}} \forall y^{\mathbb{N}}.\ \mathsf{P}_a xy \to (\exists z^{\mathbb{N}} (x,z) \in X \wedge \mathsf{P}_a xz)$$

and

$$\mathsf{fun}(X) := \forall x^{\mathbb{N}} \exists y^{\mathbb{N}}.\ (x,y) \in X \wedge \forall z^{\mathbb{N}}.\ (x,z) \in X \to y = z.$$

Indeed, as a witness for X, we may take the formula

$$S(n,m) := \left(\mathsf{P}_a nm \wedge \forall x^{\mathbb{N}}.\, x \leqslant m \to \neg \mathsf{P}_a nx\right) \vee \left(\forall y^{\mathbb{N}} \neg \mathsf{P}_a ny \wedge m = 0\right).$$

Then, using EM_1, we can prove $A[S/X]$ and thus $\exists XA$.

However, such an approach is fairly inefficient because it requires us to *define* a particular Skolem function, and it usually ends by defining a map always returning the minimum witness, as in the example, or by imposing some arbitrary criterion on the possible witnesses.

Instead, our approach of adding SK_1 is more direct and efficient: our realizer of SK_1 will not waste resources by always constructing a minimum witness, but it will keep the first one it finds. The addition of non-computable functions to the language of arithmetic in fact requires a non-trivial modification of the realizability semantics. For example, there is no formulation of Krivine classical realizability (Krivine 2009) realizing the formula SK_1 as it is written. This is due to the fact that in order to represent arithmetic in pure second-order logic, we need to relativise all the first-order quantifiers to the usual predicate $\mathsf{Nat}(x)$, which defines the concept of natural number. This is not enough, though, because we must then ensure that for every first-order term t in the language, it is possible to prove $\mathsf{Nat}(t)$. There is no problem in the case of a term language with function symbols representing recursive functions, but if Φ is added, there would be the axiom $\forall x.\mathsf{Nat}(x) \to \mathsf{Nat}(\Phi x)$. But then, by the standard results on storage operators showing that it is always possible to effectively recover the number denoted by t from a realizer of $\mathsf{Nat}(t)$, we would get that Φ is computable if the axiom is realizable.

3.1. *Language of* $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$

We now define the language of the arithmetical theory $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$.

Definition 3.2 (language of $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$). The language $\mathcal{L}_{\text{Class}}$ of $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$ is defined as follows:

(1) The terms of $\mathcal{L}_{\text{Class}}$ are all $t \in \mathcal{T}_{\text{Class}}$ such that $t : \mathtt{N}$ and

$$\mathsf{FreeVar}(t) \subseteq \{x_1^{\mathbb{N}}, \ldots, x_n^{\mathbb{N}}\}$$

for some $x_1, \ldots, x_n$.

(2) The atomic formulas of $\mathcal{L}_{\text{Class}}$ are all $Q \in \mathcal{T}_{\text{Class}}$ and Xt such that $Q : \mathtt{Bool}$ and

$$\mathsf{FreeVar}(Q) \subseteq \{x_1^{\mathbb{N}}, \ldots, x_n^{\mathbb{N}}\}$$

for some $x_1, \ldots, x_n$ and X is a predicate variable.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(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, with quantifiers possibly ranging over second-order predicate variables.
(4) As usual, if A and B are formulas of $\mathcal{L}_{\text{Class}}$ and X is a set variable, we write $A[\lambda z B(z)/X]$ to denote the formula obtained from A by replacing all its atomic subformulas of the form Xt with $B[t/z]$ (without capturing any of the free variables of B).

We use $\perp$ to denote the atomic formula `False`. We shall write natural number variables in lower-case characters $x^{\mathbb{N}}, y^{\mathbb{N}}, z^{\mathbb{N}}, \alpha^{\mathbb{N}}$ and predicate variables in upper-case characters X, Y, Z. We shall often omit the types of natural number variables, writing, for instance, $\forall x\, A$ in place of $\forall x^{\mathbb{N}} A$. If P is an atomic formula of $\mathcal{L}_{\text{Class}}$ in the free variables $x_1, \ldots, x_n$ and $t_1, \ldots, t_n$ are terms of $\mathcal{L}_{\text{Class}}$, with $P(t_1, \ldots, t_n)$ we shall denote the atomic formula by $P[t_1/x_1, \ldots, t_n/x_n]$.

We defined $\Rightarrow_{\texttt{Bool}} : \texttt{Bool}, \texttt{Bool} \to \texttt{Bool}$ as a term implementing implication, therefore, to be accurate, formulas of the form $\mathsf{P}_a(t,u) \Rightarrow_{\texttt{Bool}} \mathsf{P}_a(t, \Phi_a t)$ are not an implication between two atomic formulas, but they are equal to the single atomic formula Q, where

$$Q := \Rightarrow_{\texttt{Bool}} (\mathsf{P}_a t u)(\mathsf{P}_a t(\Phi_a t)).$$

Any atomic formula A of $\mathcal{L}_{\text{Class}}$ is a boolean term of $\mathcal{T}_{\text{Class}}$, so for any state s, we may form the 'approximation' $A[s] : \texttt{Bool}$, with $A[s] \in \mathcal{F}$, of A. In $A[s]$, we replace the Skolem function Φ we have in A by its approximation s.

Our definition of realizability provides a formal semantics for $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$, and therefore also for the more usual language of arithmetic $\mathsf{HAS} + \mathsf{EM}_1$, in which all terms represent recursive maps.

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

3.2. *s-saturated sets*

We now turn to the definition of a fundamental concept: the notion of an s-saturated set of type A. The concept arises naturally if we try, as a first attempt, to define realizability for second-order formulas $\forall X A$ in terms of the realizability of all formulas $A[\lambda z B(z)/X]$. Of course, such a definition would not be well founded with respect to the logical structure of formulas. In the case of second-order Heyting arithmetic, we usually takes an extensional approach to solving this issue – see Oliva and Streicher (2008), for example, which essentially follows Krivine (1990). Since the computational meaning of a formula is determined by the set of its realizers, we replace quantification over formulas with quantification over saturated sets of lambda terms, that is, arbitrary sets of terms closed under intensional equality. Then, for quantification over predicates, we can define realizability for $\forall X A$ in terms of realizability of $A[F/X]$ for every function F from natural numbers to saturated set of terms. The idea is that F represents an abstract proposition over natural numbers.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

However, Interactive Realizability is relativised to states, that is, to terms $s : \mathtt{S}$. Consequently, the intensional equality of terms of $\mathcal{F}_{\text{Class}}$, and hence saturation, are also relativised to states. The idea is that each world/state s determines a notion of equality between terms of $\mathcal{F}_{\text{Class}}$ because during computations we replace Φ with its approximation s. That is, two terms t, u of $\mathcal{F}_{\text{Class}}$ are intensionally equal in the state s if $t[s] = u[s]$ in $\mathcal{F}$.

Definition 3.3 (s-saturated sets).

(1) Let s be any state. An *s-saturated set of type A* (A closed) is any set S of closed type-A terms of $\mathcal{F}_{\text{Class}}$ such that if $t \in S$ and $t[s] = u[s]$ in $\mathcal{F}$, then $u \in S$.
(2) For every type A and any state s, we write $\mathsf{Sat}_A(s)$ to denote the set of all *s-saturated sets of type A*.
(3) Let $\mathcal{L}^+_{\text{Class}}$ be the language resulting from $\mathcal{L}_{\text{Class}}$ by adding a predicate constant symbol $\mathscr{F}$ for every triple made up of a function F, type A and state s such that $F : \mathbb{N} \to \mathsf{Sat}_A(s)$. We use $[\![\mathscr{F}]\!]$ to denote the function F, with $|\mathscr{F}|$ the type A and with $\mathsf{st}(\mathscr{F})$ the state s.

Remark 3.4. We shall not assign boolean values to propositions of the form $\mathscr{F}n$, let alone arbitrary formulas, since we are interested in determining what the construction of a formula is rather than when a formula is true. Realizability will indeed give a meaning to the expression $\mathscr{F}n$ through the set $[\![\mathscr{F}]\!](n)$, which specifies in an extensional way what it means to 'construct' $\mathscr{F}n$ in a world/state s. More generally, the meaning of an arbitrary formula in a state s will just be a description of what a construction of it is in the state s, so it will be determined by all its possible realizers in the state s. The (unconditional) meaning of an arbitrary formula will just be the set of programs that are valid realizers of it in every state s, and if this set is not empty, the formula is considered to be 'true' in our framework.

3.3. *Interactive Realizability*

We are now going to define for every formula A of $\mathcal{L}^+_{\text{Class}}$, what type $|A|$ a realizer of A must have.

Definition 3.5 (types for realizers). For each formula A of $\mathcal{L}^+_{\text{Class}}$, we define a type $|A|$ of $\mathcal{F}_{\text{Class}}$ by induction on A:

(1) $|P| = \mathtt{U}$.
(2) $|Xt| = X$.
(3) $|\mathscr{F}t| = |\mathscr{F}|$.
(4) $|A \wedge B| = |A| \times |B|$.
(5) $|A \vee B| = \mathtt{Bool} \times (|A| + |B|)$.
(6) $|A \to B| = |A| \to |B|$.
(7) $|\forall x A| = \mathtt{N} \to |A|$.
(8) $|\forall X A| = \forall X |A|$.
(9) $|\exists x A| = \mathtt{N} \times |A|$.
(10) $|\exists X A| = \exists X |A|$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

We now define the realizability relation $t \Vdash C$ to be closed when $t \in \mathcal{F}_{\text{Class}}$, $C \in \mathcal{L}^{+}_{\text{Class}}$ is closed and $t : |C|$.

Definition 3.6 (Interactive Realizability). Assume s is a state, t is a closed term of $\mathcal{F}_{\text{Class}}$, $C \in \mathcal{L}^{+}_{\text{Class}}$ is a closed formula and $t : |C|$. We first define the relation $t \Vdash_s C$ by induction and by cases according to the form of C:

(1) $t \Vdash_s Q$ if and only if:

— $t[s] = \overline{U}$ implies that U is sound and $\mathsf{dom}(U) \cap \mathsf{dom}(s) = \varnothing$

— $t[s] = \varnothing$ implies $Q[s] = \mathtt{True}$.

(2) $t \Vdash_s \mathscr{F} u$ if and only if $\mathsf{st}(\mathscr{F}) = s$ and for some numeral n, we have $u[s] = n$ and $t \in [\![\mathscr{F}]\!](n)$.

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

(4) $t \Vdash_s A \vee B$ if and only if either

$$\begin{aligned} \pi_0 t[s] &= \mathtt{True} \\ \pi_1 t[s] &= \iota_{0,|A|,|B|}(u) \\ u &\Vdash_s A, \end{aligned}$$

or

$$\begin{aligned} \pi_0 t[s] &= \mathtt{False} \\ \pi_1 t[s] &= \iota_{1,|A|,|B|}(v) \\ v &\Vdash_s B. \end{aligned}$$

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

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

(7) $t \Vdash_s \exists x A$ if and only for some numeral n, we have $\pi_0 t[s] = n$ and $\pi_1 t \Vdash_s A[n/x]$.

(8) $t \Vdash_s \forall X A$ if and only if for every $\mathscr{F}$, if $|\mathscr{F}| : B$ and $\mathsf{st}(\mathscr{F}) = s$, then $tB \Vdash_s A[\mathscr{F}/X]$.

(9) $t \Vdash_s \exists X A$ if and only if $t = \langle B, u \rangle$ and $u \Vdash_s A[\mathscr{F}/X]$ for some $|\mathscr{F}| : B$ with $\mathsf{st}(\mathscr{F}) = s$.

We define $t \Vdash A$ if and only if for all states s, we have $t \Vdash_s A$.

The ideas lying behind the definition of $\Vdash_s$ are the following. A realizer is a term t of $\mathcal{F}_{\text{Class}}$, possibly containing the non-computable Skolem function Φ; if such a function were computable, t would be an intuitionistic realizer. Since, in general, t is not computable, we calculate its approximation $t[s]$ at state s. This 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 in the form of a counterexample invalidating the current construction of the realizer. But the realizer can always turn such a negative outcome into positive information in the form of some new piece of knowledge learned about the Skolem function Φ.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

There are two important concepts that are useful for understanding the interaction between a realizer and the environment – a realizer receives *tests* as input and produces *predictions* as output:

— *Predictions*:
- A realizer t of $A \vee B$ uses s to predict which of A or B is realizable: if $\pi_0 t[s] = \mathtt{True}$, then A is realizable, and if $\pi_0 t[s] = \mathtt{False}$, then B is realizable.
- A realizer u of $\exists x A$ uses s to compute $\pi_0 u[s] = n$ and to predict that n is a witness for $\exists x A$ (that is, that $A[n/x]$ is realizable).
- A realizer of $\exists X A$ predicts the existence of a witness $[\![\mathscr{F}]\!] : \mathbb{N} \to \mathsf{Sat}_B(s)$ for $\exists X A$ (that is, that $A[\mathscr{F}/X]$ is realizable).

— *Tests*:
- A realizer t of a universal formula $\forall x^{\mathbb{N}} A$ or $\forall X A$ takes a natural number n or constant $\mathscr{F}$ as a challenge coming from the environment to provide a construction of $A[n/x]$ or $A[\mathscr{F}/X]$, whose correctness will be tested at the end of the 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 the computation.
- A realizer of an atomic formula Q or $\mathscr{F} u$ comes after a series of predictions and challenges that have been provided to test the construction of a complex formula, and then a final test is performed. In the case of $\mathscr{F} u$, we just verify that the realizer satisfies the notion of construction embodied by the function $[\![\mathscr{F}]\!]$. In the case of a formula Q of $\mathcal{T}_{\text{Class}}$, the realizer computes the formula Q in the state s as an experiment. Since the predictions of realizers need not always be correct, it is possible that a realized atomic formula is actually false: we may have $t \Vdash_s Q$ and $Q[s] = \mathtt{False}$ in $\mathcal{F}$. If Q, though predicted to be true, is instead false, then a counterexample has been found, which means that the approximation s of Φ is still inadequate. In this case, $t[s] \neq \varnothing$ by definition of $t \Vdash_s Q$, and the atomic realizer t takes s and extends it to a larger state s', which is the union of s and $t[s]$. That is to say, the construction of a realizer is wrong in a particular state, and the realizer must learn from its mistakes. The point is that after every instance of learning, the actual state grows, and if we ask the same realizer for new predictions, we will obtain 'better' answers.

We can say more about this final point. Suppose, for instance, that $t \Vdash A \vee B$ and let $\{s_i\}_{i \in \mathbb{N}}$ be a w.i. sequence of states. Then, since $t : \mathtt{Bool} \times |A| + |B|$, we have $\pi_0 t : \mathtt{Bool}$ is a closed term of $\mathcal{F}_{\text{Class}}$, which converges in $\{s_i\}_{i \in \mathbb{N}}$ to a boolean by Theorem 2.12. Thus, the sequence of predictions $\{\pi_0 t[s_i]\}_{i \in \mathbb{N}}$ eventually stabilises, so a witness is eventually learned in the limit.

In the atomic case, in order to have $t \Vdash_s Q$, we stipulate as a second requirement that if $t[s] = \varnothing$, then $Q[s] = \mathtt{True}$ in $\mathcal{F}$, in addition to the first requirement, that is to say, if $t[s]$ contains no *new* information about Φ for correcting wrong values of s, then t must

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

ensure the truth of Q with respect to s. Hence, the search for truth will be for us the *computation of a zero* (that is, for an s such that $t[s] = \emptyset$) driven by the excluded-middle instances and the Skolem axioms used by proofs, rather than by an exhaustive search for counterexamples.

The next proposition states that realizability at state s respects the notion of equality of $\mathcal{F}_{\text{Class}}$ terms when the latter is relativised to state s. That is, if two terms are equal at the state s, then they realize the same formulas at the state s. Hence, the extensional notion of an s-saturated set comprehends the intensional notion of realizability of a formula at state s.

Proposition 3.7 (saturation). The following hold:

(1) $\{t \mid t \Vdash_s A\} \in \mathsf{Sat}_{|A|}(s)$.
(2) If $u[s] = v[s]$, then $\{t \mid t \Vdash_s B[u/x]\} = \{t \mid t \Vdash_s B[v/x]\}$.

Proof. The proof is by straightforward induction on A. □

The next proposition states that, in the context of realizability, quantification over maps from $\mathbb{N}$ to s-saturated sets definable through realizability is the same as quantification over definable sets of natural numbers. This is crucial for proving that $t \Vdash_s \forall X A$ implies $t \Vdash_s A[\lambda x B(x)/X]$ for every formula $B(x)$ in the only free variable x.

Proposition 3.8 (comprehension). Let $B(x)$ be a formula of $\mathcal{L}^+_{\text{Class}}$ in the only free natural number variable x. Let $\mathscr{B}$ be such that

$$\begin{aligned} |\mathscr{B}| &= |B| \\ \mathsf{st}(\mathscr{B}) &= s \\ [\![\mathscr{B}]\!] &= n \mapsto \{t \mid t \Vdash_s B(n)\}. \end{aligned}$$

Then for every t,

$$t \Vdash_s A[\mathscr{B}/X] \iff t \Vdash_s A[\lambda x B(x)/X].$$

Proof. We use induction on A:

(1) $A = P$, with P predicate of $\mathcal{T}_{\text{Class}}$.
Then,

$$A[\mathscr{B}/X] = A[\lambda x B(x)/X]$$

and, trivially,

$$t \Vdash_s A[\mathscr{B}/X] \iff t \Vdash_s A[\lambda x B(x)/X].$$

(2) $A = Y\,u$, with Y a predicate variable.
Then, if $Y \neq X$, the statement is trivial since we would have

$$A[\mathscr{B}/X] = Y\,u = A[\lambda x B(x)/X].$$

So let us suppose $Y = X$ and that for some numeral n, we have $u[s] = n$. Then

$$\begin{aligned} A[\mathscr{B}/X] &= \mathscr{B}u \\ A[\lambda x B(x)/X] &= B(u), \end{aligned}$$

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

so

$$\begin{aligned} t \Vdash_s A[\mathscr{B}/X] &\iff t \Vdash_s \mathscr{B}u \\ &\iff t \in [\![\mathscr{B}]\!](n) = \{v \mid v \Vdash_s B(n)\} \\ &\iff t \Vdash_s B(n) \\ &\iff t \Vdash_s B(u) = A[\lambda x B(x)/X] \qquad \text{(by Proposition 3.7).} \end{aligned}$$

(3) The other cases are straightforward. □

Example 3.9. The most remarkable feature of our realizability semantics is the existence of a realizer for EM_1. Assume that P_a is a predicate of T and define

$$\mathsf{E}_a := \lambda\alpha^{\mathtt{N}}\langle \mathsf{X}_a\alpha,\ \mathsf{if}\ \mathsf{X}_a\alpha\ \mathsf{then}\ \iota_{0,A,B}(\langle \Phi_a\alpha,\ \emptyset\rangle)\ \mathsf{else}\ \iota_{1,A,B}(\lambda n^{\mathtt{N}}\ \mathsf{if}\ \mathsf{P}_a\alpha n\ \mathsf{then}\ \mathsf{mkupd}\, a\, \alpha\, n\ \mathsf{else}\ \emptyset)\rangle$$

with

$$\begin{aligned} A &= \mathtt{N} \times \mathtt{U} \\ B &= \mathtt{N} \to \mathtt{U}. \end{aligned}$$

Indeed, as we will explain in a moment, E_a realizes its associated instance of EM_1.

Proposition 3.10 (realizer E_a of EM_1).

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

Proof. See Aschieri and Berardi (2012). □

E_a works by using the oracle X_a (see Definition 2.10) to make predictions about which of $\exists y\, \mathsf{P}_a(m,y)$ or $\forall y \neg_{\mathtt{Bool}} \mathsf{P}_a(m,y)$ is true. Then, X_a relies, in turn, on the approximation s of Φ to make its own prediction. If $\mathsf{X}_a m[s] = \mathtt{False}$, given any n, we have $\neg_{\mathtt{Bool}}\mathsf{P}_a(m,n)$ is predicted to be true, otherwise, we have a counterexample and $\mathsf{mkupd}\, a\, m\, n$ returns $\{(a,m,n)\}$, that is, it requires a correction to the state s so as to output n on input (a,m). On the other hand, if $\mathsf{X}_a m[s] = \mathtt{True}$, there is unquestionable evidence that $\exists y\, \mathsf{P}_a(m,y)$ holds: namely, $\mathsf{P}_a m s_a(m) = \mathtt{True}$ by Definition 2.10 of X, so $\Phi_a m[s] = s(m)$ is computed and returned.

This is the basic mechanism through which learning is implemented: every state extension is linked with an assumption about an instance of EM_1 that has been used and turned out to be wrong (this is the only way to come across a counterexample), and in the following computations, the actual state will be bigger, and the realizer will not make the same error, and hence will be 'wiser'.

3.4. *Curry–Howard correspondence for* $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$

In Figures 2a and 2b, we define a standard natural deduction system for $\mathsf{HAS}+\mathsf{EM}_1+\mathsf{SK}_1$ (see Sorensen and Urzyczyn (2006), for example) together with a term assignment in the spirit of the Curry–Howard correspondence for classical logic.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Contexts

We use Γ to denote contexts of the form $x_1 : A_1, \ldots, x_n : A_n$, with $x_1, \ldots, x_n$ proof variables and $A_1, \ldots, A_n$ formulas of $\mathcal{L}_{\text{Class}}$.

Axioms

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

Conjunction

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

Implication

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

Disjunction introduction

$$\frac{\Gamma \vdash u : A}{\Gamma \vdash \langle \mathtt{True}, \iota_{0,|A|,|B|}(u)\rangle : A \vee B} \qquad \frac{\Gamma \vdash u : A}{\Gamma \vdash \langle \mathtt{False}, \iota_{1,|A|,|B|}(u)\rangle : A \vee B}$$

Disjunction elimination

$$\frac{\Gamma \vdash u : A \vee B \quad \Gamma \vdash w_1 : A \to C \quad \Gamma \vdash w_2 : B \to C}{\Gamma \vdash (\pi_1 u)|C|w_1 w_2 : C}$$

Universal quantification (1)

$$\frac{\Gamma \vdash u : \forall \alpha^{\mathbb{N}} A}{\Gamma \vdash ut : A[t/\alpha^{\mathbb{N}}]} \qquad \frac{\Gamma \vdash u : A}{\Gamma \vdash \lambda \alpha^{\mathbb{N}} u : \forall \alpha^{\mathbb{N}} A}$$

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

Existential quantification (1)

$$\frac{\Gamma \vdash u : A[t/\alpha^{\mathbb{N}}]}{\Gamma \vdash \langle t, u\rangle : \exists \alpha^{\mathbb{N}} A} \qquad \frac{\Gamma \vdash u : \exists \alpha^{\mathbb{N}} A \quad \Gamma \vdash t : \forall \alpha^{\mathbb{N}}. A \to C}{\Gamma \vdash t(\pi_0 u)(\pi_1 u) : C}$$

where, in the second rule $\alpha^{\mathbb{N}}$ is not free in C or in any formula B occurring in Γ.

Universal quantification (2)

$$\frac{\Gamma \vdash u : \forall X A}{\Gamma \vdash u|B| : A[\lambda x B(x)/X]} \qquad \frac{\Gamma \vdash u : A}{\Gamma \vdash \Lambda X u : \forall X A}$$

where $B(x)$ is a formula of $\mathcal{L}_{\text{Class}}$ and in the second rule X does not occur free in any formula occurring in Γ.

Existential quantification (2)

$$\frac{\Gamma \vdash u : A[\lambda x B(x)/X]}{\Gamma \vdash \langle |B|, u\rangle : \exists X A} \qquad \frac{\Gamma \vdash u : \exists X A \quad \Gamma \vdash t : \forall X. A \to C}{\Gamma \vdash u|C|t : C}$$

where, in the second rule, X is not free in C or in any formula occurring in Γ.

Fig. 2a. Term assignment rules for $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Induction

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

Post rules

$$\frac{\Gamma \vdash u_1 : P_1 \;\; \Gamma \vdash u_2 : P_2 \;\; \cdots \;\; \Gamma \vdash u_n : P_n}{\Gamma \vdash u_1 \Cup u_2 \Cup \cdots \Cup u_n : P}$$

where $n > 0$ and $P_1, P_2, \ldots, P_n, P$ are first-order atomic formulas of $\mathcal{L}_{\text{Class}}$, and the rule is a Post rule for equality, for a Peano axiom, for a classical propositional tautology or for booleans.

Post rules with no premises

$$\frac{}{\Gamma \vdash \varnothing : P}$$

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

EM1

$$\frac{}{\Gamma \vdash \mathsf{E}_a : \forall x.\, \exists y\; \mathsf{P}_a(x,y) \vee \forall y \neg_{\mathtt{Bool}} \mathsf{P}_a(x,y)}$$

SK1

$$\frac{}{\Gamma \vdash \lambda x^{\mathtt{N}} \lambda y^{\mathtt{N}} \mathsf{if}\ (\mathsf{P}_a x y \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a x(\Phi_a x))\ \mathsf{then}\ \varnothing\ \mathsf{else}\ (\mathsf{mkupd}\, a\, x\, y) : \forall x \forall y.\ \mathsf{P}_a(x,y) \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a(x, \Phi_a x)}$$

Fig. 2b. Term assignment rules for $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$ (continued).

We replace purely universal axioms (that is, Π^0_1 axioms) with Post rules, which are inferences of the form

$$\frac{\Gamma \vdash P_1 \;\; \Gamma \vdash P_2 \;\; \cdots \;\; \Gamma \vdash P_n}{\Gamma \vdash P}$$

where $P_1, \ldots, P_n, P$ are *first-order* atomic formulas of $\mathcal{L}_{\text{Class}}$ (that is, without set variables) such that for every substitution $\sigma = [n_1/x_1^{\mathtt{N}}, \ldots, n_k/x_k^{\mathtt{N}}\ s/\Phi]$ of numerals $n_1, \ldots, n_k$ and states s, we have $P_1\sigma = \ldots = P_n\sigma = \mathtt{True}$ implies $P\sigma = \mathtt{True}$. We now let $\mathsf{eq} : \mathtt{N}^2 \to \mathtt{Bool}$ be a term of Gödel's system T representing equality between natural numbers. Among the Post rules, we have the Peano axioms

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

the axioms of equality

$$\frac{}{\Gamma \vdash \mathsf{eq}\ x\, x} \qquad \frac{\Gamma \vdash \mathsf{eq}\, x\, y \quad \Gamma \vdash \mathsf{eq}\, y\, z}{\Gamma \vdash \mathsf{eq}\ x\, z} \qquad \frac{\Gamma \vdash P(x) \quad \Gamma \vdash \mathsf{eq}\, x\, y}{\Gamma \vdash P(y)}$$

and for every P_1, P_2 such that $P_1 = P_2$ is an equation of Gödel's system T (equivalently, P_1, P_2 have the same normal form in T), we have the rule

$$\frac{\Gamma \vdash P_1}{\Gamma \vdash P_2}.$$

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

We also have a Post rule

$$\frac{\Gamma \vdash P_1 \ \Gamma \vdash P_2 \ \cdots \ \Gamma \vdash P_n}{\Gamma \vdash P}$$

for every classical propositional tautology $P_1 \to \ldots \to P_n \to P$, where for $i = 1, \ldots, n$, P_i, P are atomic formulas obtained as a combination of other atomic formulas using the Gödel's system T boolean connectives. For example, we have the rules

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash P} \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \Rightarrow_{\mathtt{Bool}} Q} \qquad \frac{\Gamma \vdash P \wedge_{\mathtt{Bool}} Q}{\Gamma \vdash P}.$$

Finally, we have a Post rule of case reasoning for booleans. For any atomic formula P and any atomic formula $A[P]$, we have

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

The connectives $\vee_{\mathtt{Bool}}$ and $\vee$ have the same meaning, but are syntactically different: for every atomic formula P, we consider $P \vee_{\mathtt{Bool}} \neg_{\mathtt{Bool}} P$ to be an atomic formula and $P \vee \neg_{\mathtt{Bool}} P$ to be a compound formula; $P \vee_{\mathtt{Bool}} \neg_{\mathtt{Bool}} P$ is an axiom, but we can derive $\mathsf{HAS} \vdash P \vee \neg_{\mathtt{Bool}} P$ by case reasoning.

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 2.2, if

$$u_1 \Cup \cdots \Cup u_n[s] = \emptyset,$$

then

$$u_1[s] = \ldots = u_n[s] = \emptyset,$$

that is, all u_i 'have nothing to learn', in which case, each u_i must guarantee A_i to be true, so the conclusion of the Post rule is true because true premises $P_1, \ldots, P_n$ mean a true conclusion P.

Remark 3.11. Note that the full *ex-falso-quodlibet* axiom is not in our system since there are no rules or axioms to derive $\bot \to Xt$. Nevertheless, we do have ex-falso-quodlibet restricted to first-order formulas, so we cannot say we have minimal arithmetic, where $\bot$ is usually replaced by $0 = \mathsf{S}(0)$. We have omitted this axiom from the system because it is not very interesting from the computational point of view since it is usually interpreted by dummy realizers. Moreover, the ex-falso axiom is not realizable according to our semantics since there is no closed term t of type

$$|\forall X. \bot \to X0| = \forall X. \mathtt{U} \to X$$

in system $\mathcal{F}$, since otherwise $\Lambda X. tX\emptyset$ would have type $\forall X X$, which is not possible by normalisation and the argument used for proving Lemma 2.7. Furthermore, the trivial

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

translation mapping atomic formulas of the form Xt to $Xt \vee \bot$ eliminates the need for the full ex-falso axiom in the system $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$.

We will now prove our main result in this section: *viz.* that every theorem of $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$ is realizable. As usual in adequacy proofs for realizability, we will prove a stronger version of the theorem, which is suitable for proof by induction on proofs.

Theorem 3.12 (Adequacy Theorem). Suppose $\Gamma \vdash w : A$ in the system $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1$, with $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, and that the free variables of the formulas occurring in Γ and A are among $\alpha_1 : \mathtt{N}, \ldots, \alpha_k : \mathtt{N}, X_1, \ldots, X_m$. We now fix any state s, numerals $n_1, \ldots, n_k$ and assume

$$\begin{array}{cc} |\mathscr{F}_1| = B_1 & \mathsf{st}(\mathscr{F}_1) = s \\ \vdots & \vdots \\ |\mathscr{F}_m| = B_m & \mathsf{st}(\mathscr{F}_m) = s. \end{array}$$

For every formula C, let

$$\overline{C} := C[n_1/\alpha_1 \cdots n_k/\alpha_k \ \mathscr{F}_1/X_1 \cdots \mathscr{F}_m/X_m].$$

If $t_1, \ldots, t_n$ are terms such that

$$t_1 \Vdash_s \overline{A_1}, \ldots, t_n \Vdash_s \overline{A_n},$$

then

$$w[B_1/X_1 \cdots B_m/X_m][t_1/x_1^{|\overline{A_1}|} \cdots t_n/x_n^{|\overline{A_n}|} \ n_1/\alpha_1 \cdots n_k/\alpha_k] \Vdash_s \overline{A}.$$

Proof. For any term v, we write $\overline{v}$ to denote

$$v[B_1/X_1 \cdots B_m/X_m][t_1/x_1^{|\overline{A_1}|} \cdots t_n/x_n^{|\overline{A_n}|} \ n_1/\alpha_1 \cdots n_k/\alpha_k].$$

We have

$$|\overline{C}| = |C[\mathscr{F}_1/X_1 \cdots \mathscr{F}_m/X_m]| = |C|[B_1/X_1 \cdots B_m/X_m]$$

for all formulas C. We use $=$ to denote the provable equality in $\mathcal{F}_{\text{Class}}$. We now proceed by induction on w by considering the last rule in the derivation of $\Gamma \vdash w : A$:

(1) The rule for variables:
For some i, we have $w = x_i^{|A_i|}$ and $A = A_i$. So $\overline{w} = t_i \Vdash_s \overline{A_i} = \overline{A}$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(2) The $\wedge I$ rule:
We have

$$w = \langle u, t\rangle$$
$$A = B \wedge C$$
$$\Gamma \vdash u : B$$
$$\Gamma \vdash t : C.$$

Therefore, $\overline{w} = \langle \overline{u}, \overline{t}\rangle$. By the induction hypothesis,

$$\pi_0 \overline{w} = \overline{u} \Vdash_s \overline{B}$$
$$\pi_1 \overline{w} = \overline{t} \Vdash_s \overline{C},$$

so, by definition, $\overline{w} \Vdash_s \overline{B} \wedge \overline{C} = \overline{A}$.

(3) The $\wedge E$ rules (the left one say):
We have $w = \pi_0 u$ and $\Gamma \vdash u : A \wedge B$. So $\overline{w} = \pi_0 \overline{u} \Vdash_s \overline{A}$, because $\overline{u} \Vdash_s \overline{A} \wedge \overline{B}$ by the induction hypothesis.

(4) The $\rightarrow E$ rule:
We have

$$w = ut$$
$$\Gamma \vdash u : B \rightarrow A$$
$$\Gamma \vdash t : B.$$

So $\overline{w} = \overline{u}\overline{t} \Vdash_s \overline{A}$, for $\overline{u} \Vdash_s \overline{B} \rightarrow \overline{A}$ and $\overline{t} \Vdash_s \overline{B}$ by the induction hypothesis.

(5) The $\rightarrow I$ rule:
We have

$$w = \lambda x^{|B|} u$$
$$A = B \rightarrow C$$
$$\Gamma, x : B \vdash u : C.$$

Suppose now that $t \Vdash_s \overline{B}$. We have to prove that $\overline{w}t \Vdash_s \overline{C}$. By the induction hypothesis on u, we have $\overline{u} \Vdash_s \overline{C}$. By trivial equalities and the induction hypothesis,

$$\begin{aligned}\overline{w}t &= (\lambda x^{|\overline{B}|}\overline{u})t \\ &= u[B_1/X_1 \cdots B_m/X_m][t/x^{|\overline{B}|}\ t_1/x_1^{|\overline{A_1}|} \cdots t_n/x_n^{|\overline{A_n}|}\ n_1/\alpha_1 \cdots n_k/\alpha_k] \\ &\Vdash_s \overline{C}.\end{aligned}$$

Therefore, $\overline{w}t \Vdash_s \overline{C}$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(6) The $\vee I$ rules (the left one say):
We have

$$w = \langle \mathtt{True}, \iota_{0,|B|,|C|}(u)\rangle$$
$$A = B \vee C$$
$$\Gamma \vdash u : B.$$

So $\overline{w} = \langle \mathtt{True}, \iota_{0,|\overline{B}|,|\overline{C}|}(\overline{u}), \rangle$ and hence $\pi_0\overline{w} = \mathtt{True}$. We can then verify that $\overline{u} \Vdash_s \overline{B}$ with the help of the induction hypothesis.

(7) The $\vee E$ rule:
We have

$$w = (\pi_1 u)|D|w_1 w_2$$
$$\Gamma \vdash u : B \vee C$$
$$\Gamma \vdash w_1 : B \to D$$
$$\Gamma \vdash w_2 : C \to D$$
$$A = D.$$

Assume $\pi_0\overline{u}[s] = \mathtt{True}$. By the induction hypothesis, $\overline{u} \Vdash_s \overline{B} \vee \overline{C}$. Therefore,

$$\pi_1\overline{u}[s] = \iota_{0,|\overline{B}|,|\overline{C}|}(v) \tag{1}$$

and $v \Vdash_s \overline{B}$. Hence, by Definition 2.4 of $\iota_{0,|\overline{B}|,|\overline{C}|}$ and by (1),

$$\begin{aligned}\overline{w}[s] &= \iota_{0,|\overline{B}|,|\overline{C}|}(v)|\overline{D}|\overline{w}_1[s]\overline{w}_2[s]\\ &= (\Lambda X \lambda f^{|\overline{B}|\to X} \lambda g^{|\overline{C}|\to X} f v)|\overline{D}|\overline{w}_1[s]\overline{w}_2[s]\\ &= \overline{w}_1[s]v = \overline{w}_1 v[s].\end{aligned}$$

By the induction hypothesis, $\overline{w}_1 \Vdash_s \overline{B} \to \overline{D}$. Therefore, $\overline{w}_1 v \Vdash_s \overline{D}$. Thus, $\overline{w} \Vdash_s \overline{D}$ by saturation (Proposition 3.7).
Symmetrically, if $\pi_0\overline{u}[s] = \mathtt{False}$, we again obtain $\overline{w} \Vdash_s \overline{D}$.

(8) The (first order) $\forall E$ rule:
We have

$$w = ut$$
$$A = B[t/\alpha]$$
$$\Gamma \vdash u : \forall \alpha B.$$

So $\overline{w} = \overline{u}\overline{t}$. For some numeral n, we have $n = \overline{t}[s]$. By the induction hypothesis, $\overline{u} \Vdash_s \forall\alpha\overline{B}$, so $\overline{u}n \Vdash_s \overline{B}[n/\alpha]$. Since $\overline{u}\overline{t}[s] = \overline{u}n[s]$ by saturation (Proposition 3.7), we conclude that $\overline{u}\overline{t} \Vdash_s \overline{B}[\overline{t}/\alpha]$.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(9) The (first order) $\forall I$ rule:
We have

$$w = \lambda\alpha^{\mathbb{N}} u$$
$$A = \forall\alpha B$$
$$\Gamma \vdash u : B$$

(with α not occurring free in the formulas of Γ). So $\overline{w} = \lambda\alpha^{\mathbb{N}}\overline{u}$, since $\alpha \neq \alpha_1, \ldots, \alpha_n$. Let n be a numeral. We have to prove that $\overline{w}n = \overline{u}[n/\alpha] \Vdash_s \overline{B}[n/\alpha]$, which amounts to showing that the induction hypothesis can be applied to u. To do this, it is enough to observe that for $i = 1, \ldots, n$

$$t_i \Vdash_s \overline{A_i} = \overline{A_i}[n/\alpha].$$

(10) The (first order) $\exists E$ rule:
We have

$$w = t(\pi_0 u)(\pi_1 u)$$
$$\Gamma \vdash t : \forall\alpha.\, B \to A$$
$$\Gamma \vdash u : \exists\alpha\, B.$$

Assume $n = \pi_0\overline{u}[s]$, for some numeral n. By the induction hypothesis

$$\bar{t} \Vdash_s \forall\alpha.\, \overline{B} \to \overline{A}$$
$$\overline{u} \Vdash_s \exists\alpha\, \overline{B}.$$

Therefore,

$$\pi_1\overline{u} \Vdash_s \overline{B}[n/\alpha]$$
$$\bar{t}n(\pi_1\overline{u}) \Vdash_s \overline{A}[n/\alpha] = \overline{A}$$

for α not free in $\overline{A}$. Since $\overline{w}[s] = \bar{t}n(\pi_1\overline{u})[s]$, we obtain $\overline{w} \Vdash_s \overline{A}$ by saturation (Proposition 3.7).

(11) The (first order) $\exists I$ rule:
We have

$$w = \langle t, u\rangle$$
$$A = \exists\alpha B$$
$$\Gamma \vdash u : B[t/\alpha].$$

So

$$\overline{w} = \langle \bar{t}, \overline{u}\rangle$$
$$\pi_1\overline{w} = \overline{u} \Vdash_s \overline{B}[\bar{t}/\alpha]$$

by the induction hypothesis. Assume $\pi_0\overline{w}[s] = \bar{t}[s] = n$, with n numeral. We then obtain $\pi_1\overline{w} \Vdash_s \overline{B}[n/\alpha]$ by saturation (Proposition 3.7).

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

(12) The (second order) $\forall E$ rule:
We have

$$\begin{aligned} w &= u|C| \\ A &= B[\lambda x C(x)/X] \\ \Gamma &\vdash u : \forall X B. \end{aligned}$$

So $\overline{w} = \overline{u}|\overline{C}|$. Let $\mathscr{C}$ be such that

$$\begin{aligned} |\mathscr{C}| &= |\overline{C}| \\ \mathsf{st}(\mathscr{C}) &= s \\ [\![\mathscr{C}]\!] &:= n \mapsto \{t \mid t \Vdash_s \overline{C}(n)\} \end{aligned}$$

($\mathscr{C}$ is well defined by Proposition 3.7). By the induction hypothesis, $\overline{u} \Vdash_s \forall X \overline{B}$, so $\overline{u}|\overline{C}| \Vdash_s \overline{B}[\mathscr{C}/X]$. By Proposition 3.8, we can then conclude that

$$\overline{u}|\overline{C}| \Vdash_s \overline{B}[\lambda x \overline{C}(x)/X].$$

(13) The (second order) $\forall I$ rule:
We have

$$\begin{aligned} w &= \Lambda X u \\ A &= \forall X B \\ \Gamma &\vdash u : B \end{aligned}$$

(and X does not occur free in the formulas of Γ). So we have $\overline{w} = \Lambda X \overline{u}$, since $X \neq X_1, \ldots, X_m$. Suppose $|\mathscr{F}| = C$ and $\mathsf{st}(\mathscr{F}) = s$. We have to prove that

$$\overline{w}C = \overline{u}[C/X] \Vdash_s \overline{B}[\mathscr{F}/X],$$

which amounts to showing that the induction hypothesis can be applied to u. To do this, it is enough to observe that for $i = 1, \ldots, n$,

$$t_i \Vdash_s \overline{A_i} = \overline{A_i}[\mathscr{F}/X].$$

(14) The (second order) $\exists E$ rule:
We have

$$\begin{aligned} w &= u|A|t \\ \Gamma &\vdash t : \forall X. B \to A \\ \Gamma &\vdash u : \exists X B, \end{aligned}$$

with X not occurring free in A. By the induction hypothesis on u, we have $\overline{u} \Vdash_s \exists X \overline{B}$. Hence $\overline{u} = \langle C, v \rangle$ and $v \Vdash_s \overline{B}[\mathscr{F}/X]$, where $|\mathscr{F}| = C$ and $\mathsf{st}(\mathscr{F}) = s$. By the induction hypothesis on t, we have $\overline{t} \Vdash_s \forall X. \overline{B} \to \overline{A}$, and thus

$$\overline{t}Cv \Vdash_s \overline{A}[\mathscr{F}/X] = \overline{A}.$$

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Moreover,

$$\begin{aligned}\overline{w} &= \langle C, v\rangle|\overline{A}|\overline{t}\\ &= (\Lambda Y\, \lambda x^{\forall X.\,\overline{B}\to Y}\; xCv)|\overline{A}|\overline{t} \qquad \text{(by Definition 2.4)}\\ &= \overline{t}Cv.\end{aligned}$$

We thus obtain $\overline{w} \Vdash_s \overline{A}$ by saturation (Proposition 3.7).

(15) The (second order) $\exists I$ rule:

We have

$$\begin{aligned}w &= \langle |C|, u\rangle\\ A &= \exists X B\\ \Gamma &\vdash u : B[\lambda x C(x)/X].\end{aligned}$$

So, $\overline{w} = \langle |\overline{C}|, \overline{u}\rangle$. Moreover, by the induction hypothesis,

$$\overline{u} \Vdash_s \overline{B}[\lambda x \overline{C}(x)/X].$$

Let $\mathscr{F}$ be such that

$$\begin{aligned}|\mathscr{F}| &= |\overline{C}|\\ \mathsf{st}(\mathscr{F}) &= s\\ [\![\mathscr{F}]\!] &:= n \mapsto \{t \mid t \Vdash_s \overline{C}(n)\}\end{aligned}$$

($\mathscr{F}$ is well defined by Proposition 3.7). By Proposition 3.8,

$$\overline{u} \Vdash_s \overline{B}[\mathscr{F}/X],$$

which is what we set out to prove.

(16) The induction rule:

We have

$$\begin{aligned}w &= \lambda\alpha^{\mathsf{N}}\mathsf{R}|B|uv\alpha\\ A &= \forall\alpha B\\ \Gamma &\vdash u : B(0)\\ \Gamma &\vdash v : \forall\alpha.B(\alpha) \to B(\mathsf{S}(\alpha)).\end{aligned}$$

So $\overline{w} = \lambda\alpha^{\mathsf{N}}\mathsf{R}|\overline{B}|\overline{u}\overline{v}\alpha$. Now let n be a numeral. A plain induction on n shows that

$$\overline{w}n = \mathsf{R}|\overline{B}|\overline{u}\overline{v}n \Vdash_s \overline{B}[n/\alpha]$$

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 the induction hypothesis.

(17) A Post rule:

We have

$$\begin{aligned}w &= u_1 \Cup u_2 \Cup \cdots \Cup u_n\\ \Gamma &\vdash u_i : P_i.\end{aligned}$$

So, $\overline{w} = \overline{u}_1 \Cup \overline{u}_2 \Cup \cdots \Cup \overline{u}_n$. We first suppose that for $i = 1, \ldots, n$, we have $\overline{u}_i[s] = \overline{U_i}$ and $\overline{w}[s] = \overline{U}$. By the induction hypothesis, $\overline{u}_i \Vdash_s \overline{P}_i$ for $i = 1, \ldots, n$. Therefore,

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

$\mathsf{dom}(U_i) \cap \mathsf{dom}(s) = \varnothing$, and thus also $\mathsf{dom}(U) \cap \mathsf{dom}(s) = \varnothing$. Suppose now that $\overline{U} = \varnothing$, so we have to prove that $\overline{P}[s] = \mathtt{True}$. It suffices to prove that

$$\overline{P}_1[s] = \overline{P}_2[s] = \cdots = \overline{P}_n[s] = \mathtt{True}.$$

We have $\overline{U_1} = \cdots = \overline{U_n} = \varnothing$ and thus $\overline{P}_1[s] = \cdots = \overline{P}_n[s] = \mathtt{True}$ by $\overline{u}_i \Vdash_s \overline{P}_i$ for $i = 1, \ldots, n$.

(18) The excluded middle axiom EM_1:
We have $w \Vdash_s \mathsf{EM}_1$, which is just Proposition 3.10.

(19) A Φ-axiom rule:
We have

$$w = \lambda x^{\mathbb{N}} \lambda y^{\mathbb{N}} \mathsf{if}\ (\mathsf{P}_a xy \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a x(\Phi_a x))\ \mathsf{then}\ \varnothing\ \mathsf{else}\ (\mathsf{mkupd}\, a\, x\, y)$$

and

$$A = \forall x \forall y.\ \mathsf{P}_a(x, y) \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a(x, \Phi_a x).$$

Let n, m be two arbitrary numerals. We have to prove that

$$\overline{w} nm \Vdash_s \mathsf{P}_a(n, m) \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a(n, \Phi_a n).$$

There are two cases to consider:

(a) $\mathsf{P}_a(n, m) \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a(n, s_a n) = \mathtt{True}$.
In this case, $\overline{w} nm[s] = \varnothing$ and we only have to check that $\mathsf{dom}(s) \cap \mathsf{dom}(\varnothing) = \varnothing$, which is trivial.

(b) $\mathsf{P}_a(n, m) \Rightarrow_{\mathtt{Bool}} \mathsf{P}_a(n, s_a n) = \mathtt{False}$.
In this case,

$$\mathsf{P}_a nm = \mathtt{True}$$
$$\mathsf{P}_a n s_a n = \mathtt{False}.$$

Moreover,

$$\overline{w} nm[s] = \mathsf{mkupd}\, a\, n\, m = \overline{U}$$

with $U = \{(a, n, m)\}$. We have to check first that U is sound (see Definition 2.10), but this follows from $\mathsf{P}_a nm = \mathtt{True}$. We now have to verify that

$$\mathsf{dom}(s) \cap \mathsf{dom}(U) = \varnothing.$$

We have $\mathsf{dom}(U) = \{(a, n)\}$, and by Definition 2.10, $\mathsf{P}_a n s_a(n) = \mathtt{False}$ implies $(a, n) \notin \mathsf{dom}(s)$. Finally, we have to check that $\overline{U} \neq \varnothing$, which is indeed true. □

As a corollary of the Adequacy Theorem (Theorem 3.12), we now obtain our main theorem.

Theorem 3.13. If A is a closed formula such that $\mathsf{HAS} + \mathsf{EM}_1 + \mathsf{SK}_1 \vdash t : A$, then $t \Vdash A$.

4. Witness extraction with Interactive Realizability

In this section, we turn our attention to a crucial problem, which is an important test for any realizability semantics of classical arithmetic: *viz.* the witness extraction problem for

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Π^0_2-formulas. Given a realizer $t \Vdash \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} P x y$, where $P x y : \mathtt{Bool}$ is a term of Gödel's $\mathtt{T}$, we need to extract from t a non-trivial program taking as input a numeral n and yielding as output a witness for the formula $\exists y^{\mathbb{N}} P n y$ (that is, a numeral m such that $P n m = \mathtt{True}$). In the case of Interactive Realizability, the problem of computing this witness can be reduced to finding a 'zero' for a suitable term u of type $\mathtt{U}$, that is, a state $s : \mathtt{S}$ such that $u[s] = \emptyset$. Indeed, given any numeral n and state s, the following implications hold:

$$t \Vdash \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} P x y$$

$$\Longrightarrow$$

$$t \Vdash_s \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} P x y$$

$$\Longrightarrow$$

$$tn \Vdash_s \exists y^{\mathbb{N}} P n y$$

$$\Longrightarrow$$

$$\pi_0(tn)[s] = m \wedge \pi_1(tn) \Vdash_s P n m$$

$$\Longrightarrow$$

$$\pi_1(tn)[s] = \emptyset \implies P n m = \mathtt{True}.$$

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 $\exists y^{\mathbb{N}} P n y$. Intuitively, a zero for $\pi_1(tn)$ represents a sufficient amount of information to compute the required witness. In the rest of the paper, we will show how to compute such a zero. We propose the *Iterative Method*, which is very simple, easy to understand and provides an algorithm that can be directly written in pure lambda calculus. However, the proof of correctness uses the Axiom of Choice. The problems of finding an algorithm directly formalisable in system $\mathcal{F}$ and of proving its correctness constructively will be treated in a forthcoming paper.

4.1. *The Iterative Method*

If $t \Vdash \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} P x y$ and $u := \pi_0(tn)$, the interpretation of $u : \mathtt{U}$ is that of a state-extending operator. That is, given a state s, since $u[s]$ represents new information improving the approximation s of the oracle Φ, it is natural to associate with u the state-extending operator $\lambda s^{\mathsf{S}}. s \oplus u[s]$. The idea of the Iterative Method is to start from an arbitrary state and apply this state-extending operator until a zero of u is reached. Such series of state extensions represents a terminating learning process.

Theorem 4.1 (Zero Theorem). Let $P x y : \mathtt{Bool}$ be a term of Gödel's $\mathtt{T}$ and suppose $t \Vdash \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} P x y$. Let n be any numeral, define $u := \pi_1(tn)$ and let s be any state. Now define, by induction on n, a sequence $\{s_n\}_{n \in \mathbb{N}}$ of states as follows:

$$\begin{aligned} s_0 &:= s \\ s_{n+1} &:= s_n \oplus u[s_n] \\ &= \lambda x^{\mathbb{N}} \lambda y^{\mathbb{N}} \ \mathsf{if}\ (\mathsf{is}\, u[s_n]\, x\, y)\ \mathsf{then}\ (\mathsf{get}\, u[s_n]\, x\, y)\ \mathsf{else}\ s_n(x, y) \quad \text{(by Definition 2.10).} \end{aligned}$$

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Then, $\{s_n\}_{n\in\mathbb{N}}$ is weakly increasing and there exists a number k such that $u[s_k] = \varnothing$.

Proof. We first prove that $s_0, s_1, s_2, \ldots$ is a weakly increasing chain. Suppose

$$s_i(a,n) \neq s_{i+i}(a,n).$$

We have to prove that $(a,n) \in \mathsf{dom}(s_{i+1})$ and $(a,n) \notin \mathsf{dom}(s_i)$. By the definition of s_{i+1}, if it were $\mathsf{is}\, u[s_i]\, a\, n = \mathtt{False}$, we would have $s_i(a,n) = s_{i+i}(a,n)$, which gives a contradiction. Thus, $\mathsf{is}\, u[s_i]\, a\, n = \mathtt{True}$, and if we consider the update U such that $\overline{U} = u[s_i]$, we have

$$\mathsf{is}\, \overline{U}\, a\, n = \mathtt{True},$$

that is, $(a,n) \in \mathsf{dom}(U)$, and for some m, we have $(a,n,m) \in U$. If we let $l = \pi_0(tn)[s_i]$, then $u \Vdash_{s_i} Pnl$, which means that U is sound and

$$\mathsf{dom}(s_i) \cap \mathsf{dom}(U) = \varnothing.$$

From $\mathsf{dom}(s_i) \cap \mathsf{dom}(U) = \varnothing$ and $(a,n) \in \mathsf{dom}(U)$, we then obtain $(a,n) \notin \mathsf{dom}(s_i)$. From U is sound and $(a,n,m) \in U$, we obtain $\mathsf{P}_a nm = \mathtt{True}$. By definition,

$$s_{i+1}(a,n) = \mathsf{get}\, u[s_i]\, a\, n = \mathsf{get}\, \overline{U}\, a\, n = m.$$

Therefore, $s_{i+1}(a,n) = m$, and by Definition 2.10 of dom, we have $(a,n) \in \mathsf{dom}(s_{i+1})$, and we can conclude that $s_0, s_1, s_2, \ldots$ is weakly increasing.

Now, by Theorem 2.12, u converges over the chain $\{s_i\}_{i\in\mathbb{N}}$, so there exists $k \in \mathbb{N}$ such that for every $j \geqslant k$, we have $u[s_j] = u[s_k]$. By choice of k,

$$\begin{aligned} s_{k+1} \oplus u[s_{k+1}] &= (s_k \oplus u[s_k]) \oplus u[s_{k+1}] \\ &= (s_k \oplus u[s_k]) \oplus u[s_k] \\ &= s_k \oplus u[s_k] \\ &= s_{k+1}, \end{aligned}$$

so it must be that $u[s_{k+1}] = \varnothing$, which completes the proof. □

We are now able to extract from any realizer $t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy$, with $Pxy : \mathtt{Bool}$ a term of Gödel's T, a recursive map f from the set of numerals to the set of numerals such that $Pnf(n) = \mathtt{True}$ for all numerals n.

Theorem 4.2 (witness extraction via the Iterative Method). Suppose

$$t \Vdash \forall x^{\mathtt{N}} \exists y^{\mathtt{N}} Pxy,$$

where $Pxy : \mathtt{Bool}$ is a term of Gödel's T. Then, from t, we can effectively define a recursive function f from the set of numerals to the set of numerals such that for every numeral n, we have $Pn(f(n)) = \mathtt{True}$.

Proof. Let

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

v is of type $\mathtt{N} \to \mathtt{U}$. By the Zero Theorem (Theorem 4.1), there exists a recursive function zero from the set of numerals to the set of states such that $vn[\mathsf{zero}(n)] = \varnothing$ for every

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

numeral n. We define the function

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

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

$$tn \Vdash_{\mathsf{zero}(n)} \exists y^{\mathbb{N}} P n y,$$

and hence

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

that is,

$$vn[\mathsf{zero}(n)] = \emptyset \implies P n(f(n)) = \mathtt{True},$$

and therefore

$$P n(f(n)) = \mathtt{True},$$

which completes the proof □

The recursive function f of Theorem 4.2 is not directly representable in $\mathcal{F}$ since it uses unbounded iteration to compute zeros of atomic realizers, as in the proof of the Zero Theorem (Theorem 4.1). However, f is easily representable in pure lambda calculus by means of any fixed-point combinator. It is remarkable that we can write f quite directly in a purely functional language without using any control operators, unlike the case for Krivine classical realizability (Krivine 2009; Miquel 2009).

We may then wonder how backtracking is implemented in our extracted programs since control operators have precisely that function in Krivine (2009). The answer is that backtracking is implemented automatically in the iteration of the state-extending operator u of Theorem 4.1. More precisely, consider the chain $\{s_n\}_{n\in\mathbb{N}}$ defined in the statement of Theorem 4.1. When u is evaluated in s_n, and it is different from $\emptyset$, we have

$$\mathsf{dom}(s_{n+1}) = \mathsf{dom}(s_n \oplus u[s_n])$$

is strictly larger than $\mathsf{dom}(s_n)$. In particular, for some pair of numerals (i, j), there is a k such that (i, j, k) belongs to the update denoted by $u[s_n]$ and we have

$$k = s_{n+1}(i, j) \neq s_n(i, j).$$

The normalisation of $u[s_{n+1}]$ is perfectly equal to the normalisation of $u[s_n]$ up to the first point at which $s_{n+1}(i, j)$ is computed: this is the *backtracking point*. Instead of putting control operators in $u[s_n]$ to save all possible backtracking points and then jump directly to the correct one (which is discovered by reducing $u[s_n]$ to an update), the Iterative Method recomputes $u[s_{n+1}]$ from scratch. In this way, the backtracking point is re-encountered, but with a waste of resources. Again, this issue will be addressed in a forthcoming paper.

References

Aschieri, F. (2011a) *Learning, Realizability and Games in Classical Arithmetic*, Ph.D. Thesis, Queen Mary, University of London and University of Turin. (Available at `http://arxiv.org/abs/1012.4992`.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Aschieri, F. (2011b) Transfinite Update Procedures for Predicative Systems of Analysis. In: Bezem, M. (ed.) *Computer Science Logic (CSL'11) – 25th International Workshop/20th Annual Conference of the EACSL*, Leibniz International Proceedings in Informatics **12** 20–34.

Aschieri, F. (2012) A Constructive Analysis of Learning in Peano Arithmetic. *Annals of Pure and Applied Logic* **163** (11) 1448–1470.

Aschieri, F. (2013) Learning Based Realizability for HA + EM1 and 1-Backtracking Games: Soundness and Completeness. *Annals of Pure and Applied Logic* **164** (6) 591–617.

Aschieri, F. and Berardi, S. (2010) Interactive Learning-Based Realizability for Heyting Arithmetic with EM1. *Logical Methods in Computer Science* **6** (3) 19.

Aschieri, F. and Berardi, S. (2012) A New Use of Friedman's Translation: Interactive Realizability. In: Berger, U., Diener, H., Schuster, P. and Seisenberger, M. (eds.) *Logic, Construction, Computation*, Ontos-Verlag Series in Mathematical Logic 11–49.

Avigad, J. (2000) A Realizability Interpretation for Classical Arithmetic. In: Buss, S. R., Hájek, P. and Pudlák, P. (eds.) *Logic Colloquium '98*, Lecture Notes in Logic **13**, AK Peters 57–90.

Avigad, J. (2002) Update Procedures and 1-Consistency of Arithmetic. *Mathematical Logic Quarterly* **48** (1) 3–13.

Avigad, J. (2003) Eliminating Definitions and Skolem functions in First-Order Logic. *ACM Transactions on Computational Logic* **4** (3).

Berardi, S. and de'Liguoro, U. (2008) A Calculus of Realizers for EM1 Arithmetic. In: Kaminski, M. and Martini, S. (eds.) Computer Science Logic: Proceedings 22nd International Workshop, CSL 2008, at 17th Annual Conference of the EACSL. *Springer-Verlag Lecture Notes in Computer Science* **5213** 215–229.

Berardi, S., Bezem, M. and Coquand, T. (1998) On the Computational Content of the Axiom of Choice. *Journal of Symbolic Logic* **63** (2) 600–622.

Berger, U. (2005) Strong Normalization for Applied Lambda Calculi. *Logical Methods in Computer Science* **1** (2-3) 1–14.

Coquand, T. (1991) A Semantics of Evidence for Classical Arithmetic. In: Informal Proceedings of the Second Workshop on Logical Frameworks 87–99.

Coquand, T. (1995) A Semantics of Evidence for Classical Arithmetic. *Journal of Symbolic Logic* **60** 325–337.

Friedman, H. (1978) Classically and Intuitionistically Provable Recursive Functions. *Springer-Verlag Lecture Notes in Mathematics* **669** 21–27.

Gentzen, G. (1935a) Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen* **112** 493–565.

Gentzen, G. (1935b) Untersuchungen fiber das logische Schliessen. *Mathematische Zeitschrift* **39** 176–210, 405–431.

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

Gödel, K. (1958) Uber eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes. *Dialectica* **12** (3-4) 280–287.

Kohlenbach, U. (2008) *Applied Proof Theory*, Springer-Verlag.

Kreisel, G. (1951) On the interpretation of non-finitist proofs, part I. *Journal of Symbolic Logic* **16** 241–267.

Kreisel, G. (1959) Interpretation of analysis by means of constructive functionals of finite types. In: Heyting, A. (ed.) *Constructivity in Mathematics*, North-Holland 101–128.

Krivine, J.-L. (1990) *Lambda-calcul, types et modèles*, Studies in Logic and Foundations of Mathematics, Masson 1–176.

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press

Krivine, J.-L. (2009) Realizability in Classical Logic. In: Curien, P.-L., Herbelin, H., Krivine, J.-L. and Melliès, P.-A. (authors) *Interactive models of computation and program behavior*, Panoramas et synthèses **27**, Société Mathématique de France 197–229.

Mints, G., Tupailo, S. and Bucholz, W. (1996) Epsilon Substitution Method for Elementary Analysis. *Archive for Mathematical Logic* **35** (2) 103–130.

Miquel, A. (2009) Relating classical realizability and negative translation for existential witness extraction. In: Curien, P.-L. (ed.) Typed Lambda Calculi and Applications: Proceedings 9th International Conference, TLCA 2009. *Springer-Verlag Lecture Notes in Computer Science* **5608** 188–202.

Oliva, P. and Striecher, T. (2008) On Krivine Realizability Interpretation of Second-Order Classical Arithmetic. *Fundamenta Informaticae* **84** (2) 207–220.

Schwichtenberg, H. and Troelstra, A. (1996) *Basic Proof Theory*, Cambridge University Press.

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

Spector, C. (1962) Provably Recursive Functionals of Analysis: a Consistency Proof of Analysis by an Extension of Principles in Current Intuitionistic Mathematics. In: Dekker, J. (ed.) *Recursive Function Theory: Proceedings of Symposia in Pure Mathematics* **5**, AMS 1–27.

Troelstra, A. (1973) Metamathematical Investigations of Intuitionistic Arithmetic and Analysis. *Springer-Verlag Lecture Notes in Mathematics* **344**.

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

https://doi.org/10.1017/S0960129513000455 Published online by Cambridge University Press