

BAR RECURSION AND PRODUCTS OF SELECTION FUNCTIONS

MARTÍN ESCARDÓ AND PAULO OLIVA

Abstract. We show how two iterated products of selection functions can both be used in conjunction with system T to interpret, via the dialectica interpretation and modified realizability, full classical analysis. We also show that one iterated product is equivalent over system T to Spector's bar recursion, whereas the other is T -equivalent to modified bar recursion. Modified bar recursion itself is shown to arise directly from the iteration of a different binary product of 'skewed' selection functions. Iterations of the dependent binary products are also considered but in all cases are shown to be T -equivalent to the iteration of the simple products.

§1. Introduction. Gödel's [13] so-called dialectica interpretation reduces the consistency of Peano arithmetic to the consistency of a quantifier-free calculus of functionals T . In order to extend Gödel's interpretation to full classical analysis $PA^\omega + CA$, Spector [18] made use of the fact that $PA^\omega + CA$ can be embedded, via the negative translation, into $HA^\omega + AC_{\mathbb{N}} + DNS$. Here PA^ω and HA^ω denote Peano and Heyting arithmetic, respectively, formulated in the language of finite types, and

$$CA : \exists f^{\mathbb{N} \rightarrow \mathbb{B}} \forall n^{\mathbb{N}} (f(n) \leftrightarrow A(n))$$

is full comprehension,

$$AC_{\mathbb{N}} : \forall n^{\mathbb{N}} \exists x^X A(n, x) \rightarrow \exists f \forall n A(n, fn)$$

is countable choice, and

$$DNS : \forall n^{\mathbb{N}} \neg \neg B(n) \rightarrow \neg \neg \forall n B(n),$$

is the double negation shift, with $A(n)$ and $A(n, x)$ standing for arbitrary formulas, and $B(n) \equiv \exists x \neg A(n, x)$. Since $HA^\omega + AC_{\mathbb{N}}$, excluding the double negation shift, has a very straightforward (modified) realizability interpretation [19], as well as a dialectica interpretation [1, 13], the remaining challenge is to give a computational interpretation for DNS.

A computational interpretation of DNS was first given by Spector [18], via the dialectica interpretation. Spector devised a form of recursion on well-founded trees, nowadays known as *bar recursion*, and showed that the dialectica interpretation of DNS can be witnessed by such kind of recursion. A computational interpretation of DNS via realizability only came recently, first in [2], via a non-standard form of realizability, and then in [4, 5], via Kreisel's modified realizability. The realizability interpretation of DNS makes use of a new form of bar recursion, termed *modified bar recursion*.

It has been shown in [5] that Spector’s bar recursion is definable in system T extended with modified bar recursion, but not conversely, since Spector’s bar recursion is S1-S9 computable the model of total continuous functionals, but modified bar recursion is not.

In the present paper we revisit these functional interpretations of classical analysis from the perspective of the newly developed theory of selection functions [8, 9, 10, 11]. *Selection functionals* are functionals of type $(X \rightarrow R) \rightarrow X$, for arbitrary finite types X, R . We think of mappings $p: X \rightarrow R$ as generalised predicates, and of functionals $\varepsilon: (X \rightarrow R) \rightarrow X$ as witnessing, when possible, the “non-emptiness” of any given such predicate. For instance, if $R = \mathbb{B}$ is the set of booleans, Hilbert’s ε -constant can be viewed as a selection function. Just as ε -terms in Hilbert’s calculus can be used to define the existential quantifier, so can any selection function be used to define a *generalised quantifier* $\phi: (X \rightarrow R) \rightarrow R$ as

$$\phi(p) \stackrel{R}{=} p(\varepsilon(p)).$$

Moreover, just like the usual quantifiers \exists^X and \forall^Y can be nested to produce a quantifier on the product space $X \times Y$, so can generalised quantifiers and selection functions. We prefer to think about the nesting of selection functions (and quantifiers) as a *product operation*, since it transform selection functions over spaces X and Y into a new selection function on the product space $X \times Y$ (cf. [11]).

In this article we define two different iterations of the binary product of selection functions, one which we call *implicitly controlled* and the other which we call *explicitly controlled*. We show that modified bar recursion is T -equivalent to the implicitly controlled product of selection functions, whereas Spector’s bar recursion is T -equivalent to the explicitly controlled product of selection functions. Moreover, we also show how the two different products can be used to interpret DNS directly via modified realizability and the dialectica interpretation, respectively.

§2. Preliminaries. Before we present our main results, let us first define the formal systems used, and give an introduction to our recent work on selection functions.

2.1. Heyting arithmetic and system T . We work with a definitional extension of the finite types, allowing for infinite sequence of elements with different types. The main reason for this is to improve legibility of our constructions, as working with infinite sequence of a single type would obscure the shuffling of elements. This is formally defined as follows:

DEFINITION 2.1 (Finite types). *The set of all finite types \mathcal{T} and the sequences of finite types \mathcal{T}_s are simultaneously defined inductively as*

- \mathbb{B} (booleans) and \mathbb{N} (integers) are in \mathcal{T}
- If X and Y are in \mathcal{T} then $X \times Y$ (product) and $X \rightarrow Y$ (functions) are in \mathcal{T}
- If X is in \mathcal{T} then X^* (finite sequence) is in \mathcal{T}
- If $(X_i)_{i \in \mathbb{N}}$ is in \mathcal{T}_s then $\Pi_{i \in \mathbb{N}} X_i$ is in \mathcal{T}

and also

- If X is in \mathcal{T} then $(X)_{i \in \mathbb{N}}$ (constant sequence) is in \mathcal{T}_s
- If $(X_i)_{i \in \mathbb{N}}$ and $(Y_i)_{i \in \mathbb{N}}$ are in \mathcal{T}_s then $(X_i \times Y_i)_{i \in \mathbb{N}}$ and $(X_i \rightarrow Y_i)_{i \in \mathbb{N}}$ are in \mathcal{T}_s
- If $(X_i)_{i \in \mathbb{N}}$ is in \mathcal{T}_s then $(X_i^*)_{i \in \mathbb{N}}$ is in \mathcal{T}_s
- If $(X_i)_{i \in \mathbb{N}}$ is in \mathcal{T}_s then $(\Pi_{i < n} X_i)_{n \in \mathbb{N}}$ is in \mathcal{T}_s
- If $(X_i)_{i \in \mathbb{N}}$ is in \mathcal{T}_s then $(X_{i+c})_{i \in \mathbb{N}}$ is in \mathcal{T}_s , where $c: \mathbb{N}$.

We also assume to have a unit type \mathbb{I} so that $\Pi_{i < 0} X_i = \mathbb{I}$. We use X, Y, Z for variables ranging over the elements of \mathcal{T} . We often write $\Pi_i X_i$ for $\Pi_{i \in \mathbb{N}} X_i$, and also $\Pi_{i \geq k} X_i$ for $\Pi_i X_{i+k}$.

Let HA^ω be usual Heyting arithmetic in all finite types, extended to sequence types as well, with a fully extensional treatment of equality, as in the system E-HA^ω of [19]. Its quantifier-free fragment is the usual Gödel's system T , also extended with sequence types. Gödel's primitive recursion for each sequence of types $(X_i)_{i \in \mathbb{N}} \in \mathcal{T}_s$ is given by

$$\begin{aligned} \text{Rfg}0 & \quad \stackrel{X_0}{=} \quad g \\ \text{Rfg}(n+1) & \quad \stackrel{X_{n+1}}{=} \quad fn(\text{Rfg}n) \end{aligned}$$

where R has finite type $\Pi_n (X_n \rightarrow X_{n+1}) \rightarrow X_0 \rightarrow \Pi_i X_i$. If the reader prefers, however, she can assume that all X_i are equal X and read $\Pi_{i \in \mathbb{N}} X_i$ as X^ω . We also assume that we have a constant $\mathbf{0}^X$ of each finite type X , and the usual constructors and destructors such as $\langle t^X, s^Y \rangle: X \times Y$ and $\pi_i(\langle s_0^{X_0}, s_1^{X_1} \rangle) = s_i$, where $i = \{0, 1\}$, for instance. For the newly introduced sequence types we have that if $t: \Pi_i X_i$ then $ti: X_i$; and if $t: X_i$ then $\lambda i.t: \Pi_i X_i$. If $s: \Pi_{i < n} X_i$, we write $s_i: X_i$ for the i -th element of the sequence, for $i < n$. If $s: \Pi_{i < n} (X_i \times Y_i)$ is a sequence of pairs, we write $s^0: \Pi_{i < n} X_i$ and $s^1: \Pi_{i < n} Y_i$ for the projection of the sequence on the first and second coordinates, respectively. If α has type $\Pi_{i \in \mathbb{N}} X_i$ we use the following abbreviations

$$\begin{aligned} \alpha[k, n] & \equiv \langle \alpha(k), \dots, \alpha(n) \rangle, \quad (\text{finite segment from position } k \text{ to } n) \\ [\alpha](n) & \equiv \alpha[0, n-1], \quad (\text{initial segment of } \alpha \text{ of length } n) \\ \overline{\alpha, \bar{n}} & \equiv \langle \alpha(0), \dots, \alpha(n-1), \mathbf{0}, \mathbf{0}, \dots \rangle, \quad (\text{infinite extension of } [\alpha](n) \text{ with } \mathbf{0}'\text{s}) \end{aligned}$$

where in the last case the type of $\mathbf{0}$ at the i -th coordinate is the same type of $\alpha(i)$. If x has type X_n and s has type $\Pi_{i < n} X_i$ then $s * x$ is the concatenation of s with x , which has type $\Pi_{i < n+1} X_i$. Similarly, if x has type X_0 and α has type $\Pi_i X_{i+1}$ then $x * \alpha$ has type $\Pi_i X_i$.

In the following we shall assume that certain types are *discrete*. Semantically, in the model of total continuous functionals, discreteness means that singletons are open or that all points are isolated. Syntactically, the following grammar produces discrete types in that model (along with compact types) [8].

DEFINITION 2.2 (Discrete and compact types). *Define the two subsets of \mathcal{T} inductively as follows:*

$$\begin{aligned} \text{compact} & ::= \mathbb{B} \mid \text{compact} \times \text{compact} \mid \text{discrete} \rightarrow \text{compact} \\ \text{discrete} & ::= \mathbb{B} \mid \mathbb{N} \mid \text{discrete} \times \text{discrete} \mid \text{discrete}^* \mid \text{compact} \rightarrow \text{discrete}. \end{aligned}$$

For the first part of the paper, up to the end of Section 5, we work with a model independent notion of definability. Formally, given a term t in system T , we view an equation $F(x) = t(F, x)$ as *defining* or *specifying* a functional F . We do not worry whether such an equation has a solution in any particular model of HA^ω , or whether it is unique, when it has a solution. After this general model-independent development we consider particular models in the final section, and prove some non-definability results. The two main models we will consider are that of partial/total continuous functionals [17], and strongly majorizable functionals [7].

DEFINITION 2.3. *We say that a functional G is T -definable from a functional F (written $G \leq_T F$) over a theory \mathcal{S} if there exists a term s in system T such that $s(F)$ satisfies the defining equation of G provably in \mathcal{S} . We say that F and G are T -equivalent over \mathcal{S} , written $F =_T G$, if $G \geq_T F$ and $F \geq_T G$.*

When stating in a theorem or proposition that G is T -definable in F , we will explicitly write after the theorem/proposition number the theory \mathcal{S} that is need for the verification. In most cases this theory will be an extension of HA^ω with the following two principles: the *axiom of continuity*

$$\text{CONT} : \forall q^{\Pi_i X_i \rightarrow R} \forall \alpha \exists n \forall \beta ([\alpha](n) \stackrel{X_n}{\equiv} [\beta](n) \rightarrow q(\alpha) \stackrel{R}{=} q(\beta))$$

with R discrete, and the scheme of *relativised quantifier-free bar induction* BI

$$\left\{ \begin{array}{l} S(\langle \rangle) \\ \wedge \\ \forall \alpha \in S \exists n P([\alpha](n)) \\ \wedge \\ \forall s \in S (\forall x [S(s * x) \rightarrow P(s * x)] \rightarrow P(s)) \end{array} \right\} \rightarrow P(\langle \rangle),$$

where $S(s)$ is an arbitrary predicate, $P(s)$ a quantifier free predicate in the language of HA^ω , and $\alpha \in S$ and $s \in S$ are shorthands for $\forall n S([\alpha](n))$ and $S(s)$ respectively. For CONT it is essential that the type R is assumed to be discrete. In fact, in most cases we will only use the following consequence of BI + CONT:

LEMMA 2.4 (Continuous bar induction C-BI). *Let $(X_i)_{i \in \mathbb{N}}$ be a sequence of finite types and R a discrete finite type. Let $A_n(q)$ be a formula of HA^ω with*

$$q : \left(\prod_i X_{i+n} \rightarrow R \right) \equiv Y_n.$$

The following holds in $\text{HA}^\omega + \text{BI} + \text{CONT}$. If for all n ,

- (i) *for all c^R we have $A_n(\lambda \alpha. c)$, and*
- (ii) *for all q^{Y_n} such that $\forall x^{X_n} A_{n+1}(q_x)$ we also have $A_n(q)$,*

then for all $n^{\mathbb{N}}$ and for all q^{Y_n} we have $A_n(q)$.

PROOF. Let a formula $A(q)$ be fixed, and assume (i) and (ii). Given $n : \mathbb{N}$ and $q : Y_n$, define

$$P(s) = A_{n+|s|}(q_s)$$

and $S(s) = \text{true}$. It is clear that $P(\langle \rangle)$ implies the desired conclusion. We prove $P(\langle \rangle)$ by BI. First, let us show that $\forall \alpha \exists k P([\alpha](k))$. Given $\alpha : \Pi_{i+n} X_i$ let k the

point of continuity of q on α , using CONT. That implies that $q_{[\alpha](k)}$ is a constant function, and by (i) we have $A_{n+k}(q_{[\alpha](k)})$. We must now show $\forall s(\forall x P(s * x) \rightarrow P(s))$. Unfolding the definition of $P(s)$ we this is the same as

$$\forall s(\forall x A_{n+|s|+1}(q_{s*x}) \rightarrow A_{n+|s|}(q_s)).$$

Fixing s , this follows directly from (ii). \dashv

It is easy see that continuous bar induction implies CONT.

2.2. Selection functions and generalised quantifiers. In [11] we have studied the properties of functionals having the type $(X \rightarrow R) \rightarrow R$, and called these *generalised quantifiers*. When $R = \mathbb{B}$ we have that $(X \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$ is the type of the usual logical quantifiers \forall, \exists . We also showed that some generalised quantifiers $\phi: (X \rightarrow R) \rightarrow R$ are *attainable*, in the sense that for some *selection function* $\varepsilon: (X \rightarrow R) \rightarrow X$, we have

$$\phi p = p(\varepsilon p)$$

for all (generalised) predicates p . In the case when ϕ is the usual existential quantifier, for instance, ε corresponds to Hilbert's epsilon term. Since the types $(X \rightarrow R) \rightarrow R$ and $(X \rightarrow R) \rightarrow X$ will be used quite often, we abbreviate them as $K_R X$ and $J_R X$, respectively. Moreover, when R is fixed, we often simply write KX and JX , omitting the subscript R . In [11] we also defined products of quantifiers and selection functions.

DEFINITION 2.5 (Product of selection functions and quantifiers). *Given generalised quantifiers $\phi: KX$ and $\psi: KY$, define the product quantifier $(\phi \otimes \psi): K(X \times Y)$ as*

$$(\phi \otimes \psi)(p^{X \times Y \rightarrow R}) \stackrel{R}{=} \phi(\lambda x^X. \psi(\lambda y^Y. p(x, y))).$$

Also, given selection functions $\varepsilon: JX$ and $\delta: JY$, define the product selection function $(\varepsilon \otimes \delta): J(X \times Y)$ as

$$(\varepsilon \otimes \delta)(p^{X \times Y \rightarrow R}) \stackrel{X \times Y}{=} (a, b(a))$$

where

$$\begin{aligned} a & \stackrel{X}{=} \varepsilon(\lambda x^X. p(x, b(x))) \\ b(x^X) & \stackrel{Y}{=} \delta(\lambda y^Y. p(x, y)). \end{aligned}$$

One of the results we obtained is that the product of attainable quantifiers is also attainable. This follows from the fact that the product of quantifiers corresponds to the product of selection functions, as made precise in the following lemma.

LEMMA 2.6 ([11], lemma 3.1.2). *Let R be fixed. Given a selection function $\varepsilon: JX$, define a quantifier $\bar{\varepsilon}: KX$ as*

$$\bar{\varepsilon} p = p(\varepsilon p).$$

Then for $\varepsilon: JX$ and $\delta: JY$ we have $\overline{\varepsilon \otimes \delta} = \bar{\varepsilon} \otimes \bar{\delta}$.

Given a finite sequence of selection functions or quantifiers, the two binary products defined above can be iterated so as to give rise to finite products of selection functions and quantifiers. We have shown that such a construction

also appears in game theory (backward induction), algorithms (backtracking), and proof theory (interpretation of the infinite pigeon-hole principle) – see [11] for details.

In the following (Sections 3 and 4) we will describe two possible ways of iterating the binary product of selection function an infinite, or unbounded, number of times.

§3. Explicitly Controlled Product. The finite product of selection functions of Definition 2.5 can be infinitely iterated in two ways. The first, which we define in this section is via an *explicitly controlled* iteration, which we will show to correspond to Spector’s bar recursion. In the following section we also define an *implicitly controlled* iteration, which we will show to correspond to modified bar recursion.

DEFINITION 3.1 (eps). *Let $\varepsilon: \Pi_k JX_k$ be a family of selection functions. Define their explicitly controlled infinite product as*

$$(1) \quad \text{eps}_n^l(\varepsilon)(q) \stackrel{\Pi_i X_{i+n}}{=} \begin{cases} \mathbf{0} & \text{if } l(q(\mathbf{0})) < n \\ (\varepsilon_n \otimes \text{eps}_{n+1}^l(\varepsilon))(q) & \text{otherwise,} \end{cases}$$

where $q: \Pi_i X_{i+n} \rightarrow R$ and $l: R \rightarrow \mathbb{N}$. We call l the length function since it controls the length of the recursive path.

Given an infinite sequence of selection functions $\varepsilon: \Pi_k JX_{k+n}$ we repeatedly apply the binary product of selection functions to the first element of the sequence ε_n and the result of the recursive call on the tails of the sequence. That is done until the condition $l(q(\mathbf{0})) < n$ is met. In order to see why such a condition is eventually met (assuming continuity, for instance) it is best to look at the following equivalent formulation of eps.

LEMMA 3.2 (HA^ω). *Let $q: \Pi_i X_{i+n} \rightarrow R$ and $l: R \rightarrow \mathbb{N}$. The functional eps can be equivalently defined as*

$$\text{eps}_n^l(\varepsilon)(q) \stackrel{\Pi_i X_{i+n}}{=} \begin{cases} \mathbf{0} & \text{if } l(q(\mathbf{0})) < n \\ c * \text{eps}_{n+1}^l(\varepsilon)(q_c) & \text{otherwise,} \end{cases}$$

where $c = \varepsilon_n(\lambda x. \overline{\text{eps}_{n+1}^l(\varepsilon)}(q_x))$; or even more generally

$$\text{eps}_n^l(\varepsilon)(q)(i) \stackrel{X_{i+n}}{=} \begin{cases} \mathbf{0} & \text{if } A \\ \varepsilon_{i+n}(\lambda x. \overline{\text{eps}_{i+n+1}^l(\varepsilon)}(q_{t*x})) & \text{otherwise,} \end{cases}$$

where $A \equiv \exists s \preceq t(l(q_s(\mathbf{0})) < n + |s|)$ and $t = [\text{eps}_n^l(\varepsilon)(q)](i)$.

PROOF. The first equivalent formulation is obtained by simply unfolding of binary product of selection functions. For the second formulation one uses course-of-values induction noticing that as soon as the condition $l(q_s(\mathbf{0})) < n + |s|$ is satisfied for some s then the value of $\text{eps}_n^l(\varepsilon)(q)(i)$ will be $\mathbf{0}$ for $i \geq |s|$. \dashv

It is easy to see that eps exists in the model of total continuous functionals, and is in fact uniquely characterized by its defining equation. Note that the functional $\text{eps}_n(\varepsilon)(q)$ returns an infinite sequence, say $\alpha: \Pi_i X_{i+n}$. Intuitively, at

each recursive call the functional q gets information about one more element of its input sequence. If q is assume to be continuous it will eventually always return a fixed value, no matter what the rest of the input sequence is. This means that as n increases we will eventually have $l(q(\mathbf{0})) < n$. It is perhaps surprising that such a functional also exists in the model of strongly majorizable functionals [7], which contains discontinuous functionals! Following the construction of Bezem [7] one can prove this directly, but this result will also follow from our result that eps is T -definable from Spector's bar recursion (Section 3.3).

Although we only need eps for the sake of all results in this paper, it will nevertheless be useful, for the sake of clarity, to define a corresponding explicitly controlled product of quantifiers:

DEFINITION 3.3 (epq). *Let $\phi: \prod_k KX_k$ be a sequence of generalised quantifiers. Their explicitly controlled infinite product is defined as*

$$\text{epq}_n^l(\phi)(q) \stackrel{R}{=} \begin{cases} q(\mathbf{0}) & \text{if } l(q(\mathbf{0})) < n \\ (\phi_n \otimes \text{epq}_{n+1}^l(\phi))(q) & \text{otherwise,} \end{cases}$$

where $q: \prod_i X_{i+n} \rightarrow R$ and $l: R \rightarrow \mathbb{N}$. Unfolding the definition of the binary product of quantifiers we have

$$\text{epq}_n^l(\phi)(q) \stackrel{R}{=} \begin{cases} q(\mathbf{0}) & \text{if } l(q(\mathbf{0})) < n \\ \phi_n(\lambda x^{X_n}. \text{epq}_{n+1}^l(\phi)(q_x)) & \text{otherwise.} \end{cases}$$

First, we show that in terms of T -definability epq is stronger than eps . Although care has to be taken when spelling out the details, the proof essentially makes use of the fact that each selection function ε defines a quantifier, as $\phi(p) = p(\varepsilon(p))$.

PROPOSITION 3.4 ($\text{HA}^\omega + \text{C-BI}$). $\text{epq} \geq_T \text{eps}$.

PROOF. In order to define eps for the types X_i and R we shall use epq for the types X_i and $R' = \prod_i X_i$. Define

$$\text{eps}_n^l(\varepsilon)(q) \stackrel{\prod_i X_{i+n}}{=} (\text{epq}_n^l(\phi^{\varepsilon, q^n})(\lambda \alpha^{\prod_i X_{n+i}}. \mathbf{0}^{\prod_{i < n} X_i} * \alpha))^n,$$

where

$$\phi_i^{\varepsilon, q}(p^{X_i \rightarrow R'}) \stackrel{R'}{=} p(\varepsilon_i(\lambda x^{X_i}. q(p(x))))$$

and $\alpha^n(i) = \alpha(i+n)$ and $q^n(\alpha^{\prod_i X_i}) \stackrel{R}{=} q(\alpha^n)$. The construction α^n drops the first n elements of α whereas q^n behaves as q except that it ignores the first n elements of its input sequence. Assuming $l(q(\mathbf{0})) \geq n$ and that $i > 0$ (the other

case being trivial) we have

$$\begin{aligned}
\text{eps}_n^l(\varepsilon)(q)(i) &\stackrel{X_{n+i}}{=} \text{epq}_n^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0}^{\prod_{i < n} X_i} * \alpha)(n+i) \\
&= \phi_n^{\varepsilon, q^n}(\lambda x^{X_n}.\text{epq}_{n+1}^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0} * \alpha)_x)(n+i) \\
&= \phi_n^{\varepsilon, q^n}(\lambda x^{X_n}.\text{epq}_{n+1}^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0} * x * \alpha))(n+i) \\
&= \text{epq}_{n+1}^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0} * c * \alpha)(n+i) \\
&\stackrel{(+)}{=} \text{epq}_{n+1}^l(\phi^{\varepsilon, (q_c)^{n+1}})(\lambda\alpha.\mathbf{0}^{\prod_{i < n+1} X_i} * \alpha)((n+1) + (i-1)) \\
&= (\text{eps}_{n+1}^l(\varepsilon)(q_c))(i-1)
\end{aligned}$$

where

$$\begin{aligned}
c &= \varepsilon_n(\lambda x^{X_n}.q^n(\text{epq}_{n+1}^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0} * x * \alpha))) \\
&\stackrel{(+)}{=} \varepsilon_n(\lambda x^{X_n}.(q_x)^{n+1}(\text{epq}_{n+1}^l(\phi^{\varepsilon, (q_x)^{n+1}})(\lambda\alpha.\mathbf{0} * \alpha))) \\
&= \varepsilon_n(\lambda x^{X_n}.q_x(\text{epq}_{n+1}^l(\phi^{\varepsilon, (q_x)^{n+1}})(\lambda\alpha.\mathbf{0} * \alpha))^{n+1}) \\
&= \varepsilon_n(\lambda x^{X_n}.q_x(\text{eps}_{n+1}^l(\varepsilon)(q_x))).
\end{aligned}$$

Property (+) $\text{epq}_{n+1}^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0} * c * \alpha) = \text{epq}_{n+1}^l(\phi^{\varepsilon, (q_c)^{n+1}})(\lambda\alpha.\mathbf{0} * \alpha)$ follows from the definition of $\phi^{\varepsilon, q}$ by C-BI, taking

$$A_n(q) = \left(\text{epq}_{n+1}^l(\phi^{\varepsilon, q^n})(\lambda\alpha.\mathbf{0} * c * \alpha) = \text{epq}_{n+1}^l(\phi^{\varepsilon, (q_c)^{n+1}})(\lambda\alpha.\mathbf{0} * \alpha) \right).$$

Note that $q^n(\lambda\alpha.\mathbf{0} * c * \mathbf{0}) = (q_c)^{n+1}(\lambda\alpha.\mathbf{0})$, and hence $l(q^n(\lambda\alpha.\mathbf{0} * c * \mathbf{0})) = l((q_c)^{n+1}(\lambda\alpha.\mathbf{0}))$. Thus, whenever the left-hand side functional reaches the stopping condition, so does the right-hand side functional. \dashv

Hence, we have shown that eps is T -definable in epq . In the following lemma we show that epq can in turn be defined from eps , if we are working with attainable quantifiers, i.e. quantifiers that have associated selection functions.

LEMMA 3.5 ($\text{HA}^\omega + \text{C-BI}$). $\text{epq}_n^l(\bar{\varepsilon}) = \overline{\text{eps}_n^l(\varepsilon)}$.

PROOF. The above equation can also be shown by C-BI. Let

$$A_n(q) = \left(\text{epq}_n^l(\bar{\varepsilon})(q) = \overline{\text{eps}_n^l(\varepsilon)(q)} \right).$$

The case (i) when q is constant $\lambda\alpha.c$ is trivial, as $\overline{\text{eps}_n^l(\varepsilon)(q)} = c = \text{epq}_n^l(\bar{\varepsilon})(q)$. To show (ii) we fix q and assume (IH) $\forall x A_{n+1}(q_x)$. We must show $A_n(q)$. We consider two cases: If $l(q(\mathbf{0})) < n$ then

$$\text{epq}_n^l(\bar{\varepsilon})(q) = q(\mathbf{0}) = q(\text{eps}_n^l(\varepsilon)(q)) = \overline{\text{eps}_n^l(\varepsilon)(q)}.$$

If, however, $l(q(\mathbf{0})) \geq n$, then

$$\begin{aligned}
\text{epq}_n^l(\bar{\varepsilon})(q) &= (\bar{\varepsilon}_n \otimes \text{epq}_{n+1}^l(\bar{\varepsilon}))(q) \\
&= \bar{\varepsilon}_n(\lambda x.\text{epq}_{n+1}^l(\bar{\varepsilon})(q_x)) \\
&\stackrel{\text{(IH)}}{=} \bar{\varepsilon}_n(\lambda x.\overline{\text{eps}_{n+1}^l(\varepsilon)(q_x)}) \\
&\stackrel{\text{(L2.6)}}{=} \overline{(\varepsilon_n \otimes \text{eps}_{n+1}^l(\varepsilon))(q)} \equiv \overline{\text{eps}_n^l(\varepsilon)(q)}.
\end{aligned}$$

⊥

3.1. Dialectica interpretation of classical analysis. In order to find witnesses for the dialectica interpretation of DNS, and hence full classical analysis, Spector arrived at the system of equations

$$(2) \quad \begin{aligned} n & \stackrel{\mathbb{N}}{=} \omega\alpha, \\ \alpha(n) & \stackrel{\mathbb{X}}{=} \varepsilon_n(p), \\ p(\alpha(n)) & \stackrel{\mathbb{R}}{=} q\alpha, \end{aligned}$$

where $\varepsilon_n: J_R X$ and $q: (\mathbb{N} \rightarrow X) \rightarrow R$ and $\omega: (\mathbb{N} \rightarrow X) \rightarrow \mathbb{N}$ are given and $n: \mathbb{N}$ and $\alpha: \mathbb{N} \rightarrow X$ and $p: X \rightarrow R$ are the unknowns. We now show how eps can be used to solve Spector's equations. We first solve a slightly different set of equations, and as a corollary we obtain a solution to Spector's original one.

THEOREM 3.6 (HA^ω + (1)). *Let $q: \Pi_i X_i \rightarrow R$ and $l: R \rightarrow \mathbb{N}$ and $\varepsilon: \Pi_i J_R X_i$ be given. Define*

$$\begin{aligned} \alpha & = \text{eps}_0^l(\varepsilon)(q) \\ p_n(x) & = \text{epq}_{n+1}^l(\bar{\varepsilon})(q_{[\alpha](n)*x}). \end{aligned}$$

For $n \leq l(q(\alpha))$ we have

$$\begin{aligned} \alpha(n) & \stackrel{\mathbb{X}_n}{=} \varepsilon_n(p_n) \\ p_n(\alpha(n)) & \stackrel{\mathbb{R}}{=} q\alpha. \end{aligned}$$

PROOF. The proof is very similar to Spector's proof (cf. lemma 11.5 of [16]). First, let us show by induction that for all n the following holds:

$$(i) \quad \alpha = [\alpha](n) * \text{eps}_n^l(\varepsilon)(q_{[\alpha](n)}).$$

If $n = 0$ this follows by definition. Assume this holds for n , we wish to show it also holds for $n + 1$. Consider two cases.

(a) If $l(q_{[\alpha](n)}(\mathbf{0})) = l(q(\bar{\alpha}, \bar{n})) < n$ then $\text{eps}_n^l(\varepsilon)(q_{[\alpha](n)}) = \mathbf{0}$ and hence

$$\alpha \stackrel{\text{(IH)}}{=} \bar{\alpha}, \bar{n} = \overline{\alpha, n+1}.$$

Therefore, $l(q(\bar{\alpha}, \overline{n+1})) = l(q(\bar{\alpha}, \bar{n})) < n < n+1$. So,

$$[\alpha](n+1) * \text{eps}_{n+1}^l(\varepsilon)(q_{[\alpha](n+1)}) = \overline{\alpha, n+1} = \bar{\alpha}, \bar{n} = \alpha.$$

(b) If, on the other hand, $l(q_{[\alpha](n)}(\mathbf{0})) = l(q(\bar{\alpha}, \bar{n})) \geq n$, then

$$\alpha \stackrel{\text{(IH)}}{=} [\alpha](n) * \text{eps}_n^l(\varepsilon)(q_{[\alpha](n)}) = [\alpha](n) * c * \text{eps}_{n+1}^l(\varepsilon)(q_{[\alpha](n)*c}),$$

where $c = \alpha(n)$. Hence $\alpha = [\alpha](n+1) * \text{eps}_{n+1}^l(\varepsilon)(q_{[\alpha](n+1)})$.

Now, let $n \leq l(q(\alpha))$. We argue that (ii) $n \leq l(q(\bar{\alpha}, \bar{n}))$. Otherwise, assuming $n > l(q(\bar{\alpha}, \bar{n})) = l(q_{[\alpha](n)}(\mathbf{0}))$ we would have, by (i), that $\alpha = \bar{\alpha}, \bar{n}$. And hence, by extensionality, $n > l(q_{[\alpha](n)}(\mathbf{0})) = l(q(\alpha)) \geq n$, which is a contradiction.

It follows easily that, if $n \leq l(q(\alpha))$,

$$\begin{aligned}
\alpha(n) &\stackrel{(i)}{=} \text{eps}_n^l(\varepsilon)(q_{[\alpha](n)})(0) \\
&\stackrel{(ii)}{=} (\varepsilon_n \otimes \text{eps}_{n+1}^l(\varepsilon))(q_{[\alpha](n)})(0) \\
&= \varepsilon_n(\lambda x. q_{[\alpha](n)*x}(\text{eps}_{n+1}^l(\varepsilon)(q_{[\alpha](n)*x}))) \\
&= \varepsilon_n(\lambda x. \overline{\text{eps}_{n+1}^l(\varepsilon)}(q_{[\alpha](n)*x})) \\
&= \varepsilon_n(\lambda x. \text{epq}_{n+1}^l(\bar{\varepsilon})(q_{[\alpha](n)*x})) = \varepsilon_n(p_n).
\end{aligned}$$

For the second equality, we have

$$\begin{aligned}
p_n(\alpha(n)) &= \text{epq}_{n+1}^l(\bar{\varepsilon})(q_{[\alpha](n+1)}) \\
&= \overline{\text{eps}_{n+1}^l(\varepsilon)}(q_{[\alpha](n+1)}) \\
&= q_{[\alpha](n+1)}(\text{eps}_{n+1}^l(\varepsilon)(q_{[\alpha](n+1)})) \\
&= q([\alpha](n+1) * \text{eps}_{n+1}^l(\varepsilon)(q_{[\alpha](n+1)})) \stackrel{(i)}{=} q(\alpha).
\end{aligned}$$

⊣

COROLLARY 3.7. *For any given $\varepsilon_n: J_R X$ and $q: (\mathbb{N} \rightarrow X) \rightarrow R$ and $\omega: (\mathbb{N} \rightarrow X) \rightarrow \mathbb{N}$, there are $\alpha: \mathbb{N} \rightarrow X$ and $p: X \rightarrow R$ satisfying equation (2).*

PROOF. Let $R' = R \times \mathbb{N}$, and let $\pi_0: R \times \mathbb{N} \rightarrow R$ and $\pi_1: R \times \mathbb{N} \rightarrow \mathbb{N}$ denote the first and second projections. Define

$$\begin{aligned}
q'(\alpha) &\stackrel{R'}{=} \langle q(\alpha), \omega(\alpha) \rangle \\
\varepsilon'_n(p^{X \rightarrow R'}) &\stackrel{X}{=} \varepsilon_n(\lambda x^X. \pi_0(p(x))).
\end{aligned}$$

Let $\alpha = \text{eps}_0^{\pi_1}(\varepsilon')(q')$ and $p'_n(x) = \text{epq}_n^{\pi_1}(\bar{\varepsilon}')(q_{[\alpha](n)*x})$. Assume $n \leq \omega(\alpha) = \pi_1(q'(\alpha))$. By Theorem 3.6 we have

$$\begin{aligned}
\alpha(n) &\stackrel{X}{=} \varepsilon'_n(p'_n) \\
p'_n(\alpha(n)) &\stackrel{R'}{=} q'\alpha.
\end{aligned}$$

Finally, let $n = \omega(\alpha)$ and $p(x) = p_n(x) = \pi_0(p'_n(x))$. Then α and p satisfy the desired equation, since $\varepsilon'_n(p'_n) = \varepsilon_n(p_n)$. ⊣

It should be noted that we are only using the explicitly controlled product of quantifiers for the sake of clarity. As shown in Lemma 3.5, any use of epq above can be replaced by an instance of eps . Therefore, it is the recursion schema eps that solves Spector's equations. In this way, also the use of bar induction (in the proof of Lemma 3.5) is avoided. Therefore, Theorem 3.6 and Corollary 3.7 can be proven in HA^ω plus the product of selection functions, without assuming continuity or bar induction.

3.2. Dependent variant of eps . We have considered in previous papers [9, 11] a slight generalisation of the product of selection functions, where a selection function at stage n can have access to the previously computed values X_i for $i < n$. We called this the *dependent* product of selection functions and quantifiers.

DEFINITION 3.8 (Dependent product of selection functions and quantifiers). Given a quantifier $\phi: KX$ and a family of quantifiers $\psi: X \rightarrow KY$, define the dependent product quantifier $(\phi \otimes_d \psi): K(X \times Y)$ as

$$(\phi \otimes_d \psi)(p^{X \times Y \rightarrow R}) \stackrel{R}{=} \phi(\lambda x^X. \psi(x, \lambda y^Y. p(x, y))).$$

Also, given a selection function $\varepsilon: JX$ and a family of selection functions $\delta: X \rightarrow JY$, define the dependent product selection function $(\varepsilon \otimes_d \delta): J(X \times Y)$ as

$$(\varepsilon \otimes_d \delta)(p^{X \times Y \rightarrow R}) \stackrel{X \times Y}{=} (a, b(a))$$

where

$$\begin{aligned} a &\stackrel{X}{=} \varepsilon(\lambda x^X. p(x, b(x))) \\ b(x) &\stackrel{Y}{=} \delta(x, \lambda y^Y. p(x, y)). \end{aligned}$$

As done for the simple product of selection functions, we can also iterate the dependent product of selection functions as follows:

DEFINITION 3.9 (EPS). Given a family of parametrised selection functions

$$\varepsilon: \Pi_k(\Pi_{i < k} X_i \rightarrow JX_k),$$

define their dependent explicitly controlled product (denoted EPS) as

$$(3) \quad \text{EPS}_s^l(\varepsilon)(q) \stackrel{\Pi_i X_{|s|+i}}{=} \begin{cases} \mathbf{0} & \text{if } l(q(\mathbf{0})) < |s| \\ (\varepsilon_s \otimes_d (\lambda x^{X_{|s|}}. \text{EPS}_{s*x}^l(\varepsilon)))(q) & \text{otherwise.} \end{cases}$$

As done for eps in Lemma 3.2, EPS can also be equivalently formulated as

$$\text{EPS}_s^l(\varepsilon)(q)(i) \stackrel{X_{|s|+i}}{=} \begin{cases} \mathbf{0} & \text{if } A \\ \varepsilon_{s*t}(\lambda x^{X_{|s|+t}}. \overline{\text{EPS}_{s*t*x}^l(\varepsilon)}(q_{t*x})) & \text{otherwise,} \end{cases}$$

where $A \equiv \exists s \preceq t(l(q_s(\mathbf{0})) < n + |s|)$ and $t = [\text{EPS}_s^l(\varepsilon)(q)](i)$.

In the dialectica interpretation of countable choice given above (Section 3.1), the selection functions ε_n do not depend on the history of choices already made. Thus, it was sufficient to use an iteration of the *simple* product of selection functions. Nevertheless, Spector bar recursion and modified bar recursion are normally formulated in the most general form, where selection functions at point n have access to the values $i < n$.

We now show that the two iterations of the product of selection functions are T -equivalent. The idea is to make use of a construction of type

$$(X \rightarrow JY) \rightarrow J(X \rightarrow Y)$$

that turns a family of selection function into a single selection function of a (possibly) higher type level.

THEOREM 3.10 (HA $^\omega$). $\text{EPS} =_T \text{eps}$.

PROOF. It is clear that eps is T -definable from EPS. We now show how EPS is T -definable from eps. We use the alternative formulation of eps given in Lemma 3.2. A family of selection functions $\varepsilon_k: \Pi_{i=0}^{k-1} X_i \rightarrow JX_k$ can be turned into a single selection function $\tilde{\varepsilon}_k: J(\Pi_{i=0}^{k-1} X_i \rightarrow X_k)$ as

$$\tilde{\varepsilon}_k(P(\Pi_{i=0}^{k-1} X_i \rightarrow X_k) \rightarrow R) \stackrel{\Pi_{i=0}^{k-1} X_i \rightarrow X_k}{=} \lambda_S \Pi_{i=0}^{k-1} X_i . \varepsilon_S (\lambda y^{X_k} . P(\lambda t . y)).$$

Given $l: R \rightarrow \mathbb{N}$, the infinite (simple) product of selection functions gives

$$\text{eps}_0^l(\tilde{\varepsilon}) \quad : \quad J(\Pi_i(\Pi_{k < i} X_k \rightarrow X_i)).$$

The dependent product can then be defined as

$$\text{EPS}_s^l(\varepsilon)(q^{\Pi_i X_{|s+i} \rightarrow R}) \stackrel{\Pi_i X_{|s+i}}{=} (\text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s))^s,$$

where $s: \Pi_{k < j} X_k$ and $\alpha^s(i) \stackrel{X_i}{=} \alpha(i)(s * [\alpha^s](i))$ and $q^s(\alpha) = q(\alpha^s)$. We must show that EPS as defined above satisfies the defining equation (3). We do this by course-of-values induction on i . Let

$$t = [\text{EPS}_s^l(\varepsilon)(q)](i) \stackrel{(\text{IH})}{=} [(\text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s))^s](i),$$

where $q: \Pi_i X_{|s+i} \rightarrow R$ and $q^s: \Pi_k(\Pi_{i < |s|+k} X_i \rightarrow X_{|s+k}) \rightarrow R$. If for some $u \preceq t$ we have $l(q_u(\mathbf{0})) < |s| + |u|$ the result is trivial, because $(q^s)_{u'}(\mathbf{0}) = q_u(\mathbf{0})$ for some $u' \preceq t' = [\text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s)](i)$, and hence $(\text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s))^s(i) = \mathbf{0}$. On the other hand, assuming $\forall u \preceq t \ l(q_u(\mathbf{0})) \geq |s| + |u|$, unfolding definitions we have

$$\begin{aligned} \text{EPS}_s^l(\varepsilon)(q)(i) &\stackrel{X_{|s+i}}{=} (\text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s))^s(i) \\ &= \text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s)(i)(s * t) \\ &\stackrel{\text{D3.9}}{=} \tilde{\varepsilon}_{|s+i}(\lambda f . (q^s)_{u * f}(\text{eps}_{|s|+i+1}^l(\tilde{\varepsilon})((q^s)_{u * f}))) (s * t) \\ &= \varepsilon_{s * t}(\lambda x . (q^s)_{u * \lambda r . x}(\text{eps}_{|s|+i+1}^l(\tilde{\varepsilon})((q^s)_{u * \lambda r . x}))) \\ &\stackrel{(i,ii)}{=} \varepsilon_{s * t}(\lambda x . q_{t * x}(\text{eps}_{|s|+i+1}^l(\tilde{\varepsilon})((q^s)_{u * \lambda r . x}))^{s * t * x}) \\ &\stackrel{(i)}{=} \varepsilon_{s * t}(\lambda x . q_{t * x}(\text{eps}_{|s|+i+1}^l(\tilde{\varepsilon})((q_{t * x})^{s * t * x}))^{s * t * x}) \\ &= \varepsilon_{s * t}(\lambda x . q_{t * x}(\text{EPS}_{s * t * x}^l(\varepsilon)(q_{t * x}))) \\ &= \varepsilon_{s * t}(\lambda x . \text{EPS}_{s * t * x}^l(\varepsilon)(q_{t * x})), \end{aligned}$$

where $u = [\text{eps}_{|s|}^l(\tilde{\varepsilon})(q^s)](i)$. We used that

$$(i) \quad (t * \alpha)^s = t^s * (\alpha)^{s * t^s}, \text{ for all } t, s, \alpha,$$

$$(ii) \quad u^s = t, \text{ for } t, s \text{ and } u \text{ as above.}$$

–

QUESTION 3.11. *Note that a similar construction does not work in the case of quantifiers, since there is no typed λ -term of type $(X \rightarrow KY) \rightarrow K(X \rightarrow Y)$, for arbitrary X and Y . In fact, in the case of quantifiers it still open whether the simple explicitly controlled iteration is T -equivalent to the explicitly controlled iteration of the dependent product of quantifiers.*

3.3. Relation to Spector's bar recursion. As we have shown in Theorem 3.6, which is essentially Spector's solution, the explicitly controlled product of selection functions eps can also be used to give a computational interpretation of classical analysis. When presenting his solution in [18], Spector first formulates a general "construction by bar recursion" as

$$\text{BR}_s(\omega)(\phi)(q) \stackrel{R}{=} \begin{cases} q_s(\mathbf{0}) & \text{if } \omega_s(\mathbf{0}) < |s| \\ \phi_s(\lambda x^{X_{|s|}}.\text{BR}_{s*x}(\omega)(\phi)(q)) & \text{otherwise.} \end{cases}$$

This is usually referred to as *Spector's bar recursion*, but we argue that this is misleading. We show that BR is closely related to the product of *quantifiers* EPQ, whereas the special case of this used by Spector is equivalent to the (dependent) product of *selection functions* EPS, which we have shown to be equivalent to eps (Section 3.2).

REMARK 3.12. *In fact, Spector's definition seems slightly more general than BR as defined here, since in Spector's definition q might also depend on the length of the sequence s . As we show in Lemma 5.1, however, it is possible to reconstruct $|s|$ from the sequence $s * \mathbf{0}$ if s is the point where Spector's condition first happens.*

PROPOSITION 3.13 (HA^ω + C-BI). EPQ =_T BR.

PROOF. In order to define EPQ from BR, let $q^n \alpha = q(\lambda i.\alpha(i+n))$, and define

$$\text{EPQ}_s(\omega)(\phi)(q) = \text{BR}_s(\omega)(\phi)(q^{|s|}).$$

We then have (assuming $\omega_s(\mathbf{0}) \geq |s|$, the other case being trivial)

$$\begin{aligned} \text{EPQ}_s(\omega)(\phi)(q) &= \text{BR}_s(\omega)(\phi)(q^{|s|}) \\ &= \phi_s(\lambda x^{X_{|s|}}.\text{BR}_{s*x}(\omega)(\phi)(q^{|s|})) \\ &\stackrel{(+)}{=} \phi_s(\lambda x^{X_{|s|}}.\text{BR}_{s*x}(\omega)(\phi)((q_x)^{|s*x|})) \\ &= \phi_s(\lambda x^{X_{|s|}}.\text{EPQ}_{s*x}(\omega)(\phi)(q_x)) \\ &= (\phi_s \otimes \lambda x^{X_{|s|}}.\text{EPQ}_{s*x}(\omega)(\phi))(q) \end{aligned}$$

where (+) $\text{BR}_{s*x}(\omega)(\phi)(q^{|s|}) = \text{BR}_{s*x}(\omega)(\phi)((q_x)^{|s*x|})$ can, as in Proposition 3.4, be proved by C-BI, since $q^{|s|}(s * x * \alpha) = (q_x)^{|s*x|}(s * x * \alpha)$.

On the other hand, BR can be defined from EPQ as

$$\text{BR}_s(\omega)(\phi)(q) = \text{EPQ}_s(\omega)(\phi)(q_s),$$

as a simple verification shows. ⊣

Spector, however, explicitly says that only a *restricted form* of BR is used for the dialectica interpretation of (the negative translation of) countable choice. It is this restricted form that we shall from now on call *Spector's bar recursion*:

DEFINITION 3.14 (Spector's bar recursion). *Let $R = \Pi_i X_i$. Spector's bar recursion [18] is the recursion schema*

$$\text{SBR}_s(\omega)(\varepsilon) \stackrel{R}{=} \begin{cases} \hat{s} & \text{if } \omega_s(\mathbf{0}) < |s| \\ \text{SBR}_{s*c}(\omega)(\varepsilon) & \text{otherwise,} \end{cases}$$

where $c \stackrel{X_{|s|}}{=} \varepsilon_s(\lambda x^{X_{|s|}}.\text{SBR}_{s*x}(\omega)(\varepsilon))$, and where $\varepsilon_s : JX_{|s|}$ and $\omega : \Pi_i X_i \rightarrow \mathbb{N}$.

We now show that Spector's bar recursion is *T*-equivalent to the explicitly controlled product of selection functions EPS.

LEMMA 3.15 (HA^ω + C-BI). SBR ≥_T EPS.

PROOF. In order to define EPS for type R and $(X_i)_{i \in \mathbb{N}}$ we use SBR for $R' \equiv \prod_i X_i$. Let $\varepsilon_s: J_R X_{|s|}$ and $q: \prod_i X_i \rightarrow R$ and $l: R \rightarrow \mathbb{N}$ be given. Let

$$\omega^{l,q}(\alpha) = l(q(\alpha)).$$

Moreover, define the following family of selection functions $\tilde{\varepsilon}_s^q: J_{R'} X_{|s|}$

$$\tilde{\varepsilon}_s^q(f^{X_{|s|} \rightarrow R'}) \stackrel{X_{|s|}}{=} \varepsilon_s(\lambda x^{X_{|s|}}.q(f(x))).$$

For $q: \prod_i X_{i+n} \rightarrow R$ define also $q^n: \prod_i X_i \rightarrow R$ as $q^n \alpha = q(\lambda i.\alpha(i+n))$. EPS is then defined from SBR as:

$$\text{EPS}_s^l(\varepsilon)(q) \stackrel{\prod_i X_{i+|s|}}{=} \lambda i.\text{SBR}_s(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}})(|s|+i).$$

The proof is by course-of-values induction on i . Assume

$$t = [\text{EPS}_s^l(\varepsilon)(q)](i) = [\lambda i.\text{SBR}_s(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}})(|s|+i)](i).$$

We verify the case $\forall u \preceq t(l(q_u(\mathbf{0})) = \omega^{l,q^{|s|}}_u(\mathbf{0}) = \omega_u^{l,q}(\mathbf{0}) \geq |s| + |u|)$ (the other case is trivial) by course-of-values induction

$$\begin{aligned} \text{EPS}_s^l(\varepsilon)(q)(i) &\stackrel{X_{|s|+i}}{=} \text{SBR}_s(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}})(|s|+i) \\ &\stackrel{((i-1) \times)}{=} \text{SBR}_{s*t}(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}})(|s|+i) \\ &= \tilde{\varepsilon}_{s*t}^{q^{|s|}}(\lambda x.\text{SBR}_{s*t*x}(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}})) \\ &= \varepsilon_{s*t}(\lambda x.q^{|s|}(\text{SBR}_{s*t*x}(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}}))) \\ &\stackrel{(+)}{=} \varepsilon_{s*t}(\lambda x.(q_{t*x})^{|u|}(\text{SBR}_u(\omega^{l,(q_{t*x})^{|u|}})(\tilde{\varepsilon}^{(q_{t*x})^{|u|}}))) \\ &= \varepsilon_{s*t}(\lambda x.q_{t*x}(\lambda i.\text{SBR}_u(\omega^{l,(q_{t*x})^{|u|}})(\tilde{\varepsilon}^{(q_{t*x})^{|u|}})(|u|+i))) \\ &\stackrel{(\text{IH})}{=} \varepsilon_{s*t}(\lambda x.q_{t*x}(\text{EPS}_u^l(\varepsilon)(q_{t*x}))) \end{aligned}$$

where $u \equiv s * t * x$, and $((i-1) \times)$ means unfolding SBR's definition $(i-1)$ times so that $|t| = i-1$. And, for $j < i$

$$\begin{aligned} t_j &= \tilde{\varepsilon}_{s*[t](j)}^{q^{|s|}}(\lambda x.\text{SBR}_{s*[t](j)*x}(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}})) \\ &= \varepsilon_{s*[t](j)}(\lambda x.q^{|s|}(\text{SBR}_{s*[t](j)*x}(\omega^{l,q^{|s|}})(\tilde{\varepsilon}^{q^{|s|}}))) \\ &= \varepsilon_{s*[t](j)}(\lambda x.\hat{q}(\text{SBR}_{s*[t](j)*x}(\omega^{l,\hat{q}})(\tilde{\varepsilon}^{\hat{q}}))) \\ &\stackrel{(\text{IH})}{=} \varepsilon_{s*[t](j)}(\lambda x.q_{[t](j)*x}(\text{EPS}_{s*[t](j)*x}^l(\varepsilon^{\hat{q}}))), \end{aligned}$$

where $\hat{q}(\alpha) = (q_{[t](j)*x})^{|s|+j+1}(\alpha) = q^{|s|}(\alpha)$ for $\alpha = s * [t](j) * x * \beta$. Step (+) is again similar to that in the proofs of Propositions 3.4 and 3.13, and can be proven by C-BI. \dashv

PROPOSITION 3.16 (HA $^\omega$ + C-BI). $\text{EPS} =_T \text{SBR}$.

PROOF. By Lemma 3.15, it remains to show that EPS T -defines SBR. Let $R = \prod_i X_i \times \mathbb{N}$ and

$$\begin{aligned} l(r) &\stackrel{\mathbb{N}}{=} \pi_1(r) \\ q'(\alpha) &\stackrel{R}{=} \langle q(\alpha), \omega(\alpha) \rangle. \end{aligned}$$

Define

$$\text{SBR}_s(\omega)(\varepsilon) \stackrel{R}{=} s * \text{EPS}_s^l(\varepsilon)(\lambda\gamma^{\Pi_i X_{|s|+i}}.(s * \gamma)^R).$$

Assume $\omega_s(\mathbf{0}) \geq |s|$ as the other case is trivial. We then have

$$\begin{aligned} \text{SBR}_s(\omega)(\varepsilon) &= s * \text{EPS}_s^l(\varepsilon)(\lambda\gamma^{\Pi_i X_{|s|+i}}.(s * \gamma)^R) \\ &= s * c * \text{EPS}_{s*c}^l(\varepsilon)(\lambda\gamma^{\Pi_i X_{|s|+i}}.(s * c * \gamma)^R) \\ &= \text{SBR}_{s*c}(\omega)(\varepsilon) \end{aligned}$$

where $c = \varepsilon_s(\lambda x.s * x * \text{EPS}_{s*x}^l(\varepsilon)(\lambda\gamma.(s * x * \gamma))) = \varepsilon_s(\lambda x.\text{SBR}_{s*x}(\omega)(\varepsilon))$. \dashv

QUESTION 3.17. By Propositions 3.4 and 3.16, we have that

$$\text{BR} =_T \text{EPQ} \geq_T \text{SBR} =_T \text{EPS}.$$

It is open whether EPQ is T-definable from EPS (and hence from SBR) or not. We are inclined to believe this is not the case, since in EPQ we are given quantifiers, from which we might not be able to re-construct selection functions to apply EPS (see diagram in Figure 1).

§4. Implicitly Controlled Product. We have seen in Section 3 above that the explicitly controlled iterated product of selection functions is sufficient to witness the dialectica interpretation of classical countable choice. In this section we show that when interpreting this same principle via modified realizability, one seems to need an *unrestricted* or, as we shall call it, *implicitly controlled* infinite product of selection functions.

DEFINITION 4.1 (ips). The implicitly controlled product of a family $\varepsilon: \Pi_k J X_k$ of selection functions is defined as

$$\text{ips}_n(\varepsilon) \stackrel{J(\Pi_i X_{i+n})}{=} \varepsilon_n \otimes \text{ips}_{n+1}(\varepsilon).$$

We call the above infinite product *implicitly controlled* because under the assumption of continuity for functionals of type $\Pi_i X_{i+n} \rightarrow R$, the bar recursive calls eventually terminate. Unwinding the definition of the binary product, ips can also be equivalently formulated as follows.

LEMMA 4.2 (HA^ω). The functional ips can be equivalently defined by the equation

$$\text{ips}_n(\varepsilon)(q)(i) \stackrel{X_{n+i}}{=} \varepsilon_{n+i}(\lambda x^{X_{n+i}}.\overline{\text{ips}_{n+i+1}(\varepsilon)}(q_{s*x}))$$

where $q: \Pi_i X_{n+i} \rightarrow R$ and $s = [\text{ips}_n(\varepsilon)(q)](i)$.

PROOF. By unfolding the definition of the binary product of selection functions using course-of-values induction. \dashv

4.1. Realizability interpretation of classical analysis. We now describe how ips can be used to interpret countable choice via modified realizability. As discussed in the introduction, a computational interpretation of full classical analysis can be reduced to an interpretation of the double negation shift DNS. Given that the formula $A(n)$ (in DNS) can be assumed to be of the form $\exists x \neg B(n, x)$, DNS is equivalent to

$$\forall n((A(n) \rightarrow \perp) \rightarrow A(n)) \rightarrow \forall n A(n) \rightarrow \perp \rightarrow \forall n A(n).$$

That is because, for $A(n) \equiv \exists x \neg B(n, x)$, we have both $\perp \rightarrow A(n)$ and $\perp \rightarrow \forall n A(n)$ in minimal logic. Moreover, since the negative translation brings us into minimal logic, falsity \perp can be replaced by an arbitrary Σ_1^0 -formula R . (This is known as the (refined) A -translation [6], and is useful to analyse proofs of Π_2^0 theorems in analysis.) R . Recall that we are using the abbreviation

$$J_R A = (A \rightarrow R) \rightarrow A.$$

The resulting principle we obtain is what we shall call the J -shift

$$J\text{-shift} \quad : \quad \forall n J_R A(n) \rightarrow J_R \forall n A(n).$$

Thus, DNS corresponds to the K -shift, considering the other type construction $K_R X = (X \rightarrow R) \rightarrow R$. One advantage of moving to the J -shift is that $A(n)$ now can be taken to be an arbitrary formula, not necessarily of the form $\exists x \neg B(n, x)$. Hence the principle J -shift is more general than DNS. We analyse the logical strength of the principle J -shift in more detail in [10], where a proof translation based on the construction JX is also defined. Our proof of the following theorem is very similar to that of [4, Theorem 3]. We assume continuity and relativised bar induction as formulated in Section 2.1.

THEOREM 4.3 ($\text{HA}^\omega + \text{BI} + \text{CONT}$). *ips₀ modified realizes J -shift.*

PROOF. Assume that

$$\varepsilon_n \quad \text{mr} \quad (A(n) \rightarrow R) \rightarrow A(n),$$

$$q \quad \text{mr} \quad \forall n A(n) \rightarrow R.$$

We show $\forall s \in S \forall n P(s, n)$ by relativised bar induction, where

$$P(s, n) \equiv (s * \text{ips}_{|s|}(\varepsilon)(q_s))(n) \text{mr} A(n)$$

and the predicate used in the relativisation is

$$s \in S \equiv \forall n < |s| (s_n \text{mr} A(n)).$$

We write $\alpha \in S$ as an abbreviation for $\forall n ([\alpha](n) \in S)$. We now prove the first two assumptions of the bar induction, and the third assumption $S(\langle \rangle)$ is vacuously true.

(i) $\forall \alpha \in S \exists k \forall t \succeq [\alpha](k) \forall n P(t, n)$, where $t \succeq s$ means that t is an extension of the finite sequence s . Given α we pick k to be the point of continuity of q on α . The result follows simply by unfolding the definition of ips .

(ii) $\forall s \in S (\forall t, x (s * t * x \in S \rightarrow \forall n P(s * t * x, n)) \rightarrow \forall n P(s, n))$. Let $s \in S$ and assume

$$(a) \quad \forall t, x (s * t * x \in S \rightarrow \forall n P(s * t * x, n)).$$

We prove $\forall n P(s, n)$ by course-of-values induction. Assume $\forall k < n P(s, k)$, i.e.

$$(b) \quad \forall k < n ((s * \text{ips}_{|s|}(\varepsilon)(q_s))(k) \text{mr} A(k)).$$

We want to show $(s * \text{ips}_{|s|}(\varepsilon)(q_s))(n) \text{mr} A(n)$. If $n < |s|$ we are done, since in this case $(s * \text{ips}_{|s|}(\varepsilon)(q_s))(n) = s_n$ (and $s \in S$). Assume $n \geq |s|$. Then our goal becomes

$$\varepsilon_n (\lambda x^{X_n}. q_{s * t * x} (\text{ips}_{n+1}(\varepsilon)(q_{s * t * x}))) \text{mr} A(n),$$

where $t = [\text{ips}_{|s|}(\varepsilon)(q_s)](n - |s|)$. This follows from

$$\lambda x^{X_n}.q_{s*t*x}(\text{ips}_{n+1}(\varepsilon)(q_{s*t*x})) \text{mr } A(n) \rightarrow R$$

which, by definition, is

$$\forall x^{X_n}(x \text{mr } A(n) \rightarrow q_{s*t*x}(\text{ips}_{n+1}(\varepsilon)(q_{s*t*x})) \text{mr } R).$$

Let x such that $x \text{mr } A(n)$. By our assumption (b) we have that $s * t * x \in S$. And by assumption (a) we get $(s * t * x * \text{ips}_{n+1}(\varepsilon)(q_{s*t*x})) \text{mr } \forall n A(n)$. The proof is then concluded by the assumption that $q \text{mr } \forall n A(n) \rightarrow R$. \dashv

4.2. Dependent variant of ips. The proof given for the equivalence between EPS and eps in Section 3.2 can be easily adapted to show that also ips is T -equivalent to its dependent variant IPS.

DEFINITION 4.4 (IPS). Let $\varepsilon: \Pi_k(\Pi_{i<k}X_i \rightarrow JX_k)$. Define the dependent implicitly controlled product of selection functions (denoted IPS) as

$$\text{IPS}_s(\varepsilon) \stackrel{J(\Pi_i X_{|s|+i})}{=} \varepsilon_s \otimes_d (\lambda x^{X_{|s|}}.\text{IPS}_{s*x}(\varepsilon)).$$

As in Lemma 4.2 for ips, we can also equivalently define IPS as

$$\text{IPS}_s(\varepsilon)(q)(i) \stackrel{X_{|s|+i}}{=} \varepsilon_{s*t}(\lambda x^{X_{|s|+i}}.\overline{\text{IPS}_{s*t*x}(\varepsilon)}(q_{t*x})),$$

where $t = [\text{IPS}_s(\varepsilon)(q)](i)$.

We again use the fact that a family of selection functions $\Pi_{k<i}X_k \rightarrow JX_u$ can be turned into a single section function $J(\Pi_{k<i}X_k \rightarrow X_i)$ in order to simulate EPS over T using eps.

THEOREM 4.5 (HA $^\omega$ + C-BI). $\text{IPS} =_T \text{ips}$.

PROOF. It is clear that IPS is a generalisation of ips. We now show that IPS is T -definable from ips, following the same ideas used to show that EPS is T -equivalent to eps (Section 3.2). In fact, the proof here is slightly simpler since we do not have to worry about the length function l . Let $\tilde{\varepsilon}_k$ be as defined in Lemma 3.10. The infinite (simple) product of selection functions applied to $\tilde{\varepsilon}$ gives

$$\text{ips}_0(\tilde{\varepsilon}) \quad : \quad J(\Pi_i(\Pi_{k<i}X_k \rightarrow X_i)).$$

IPS can then be defined as (where $s: \Pi_{k<j}X_k$)

$$\text{IPS}_s(\varepsilon)(q^{\Pi_i X_{|s|+i} \rightarrow R}) \stackrel{\Pi_i X_{|s|+i}}{=} (\text{ips}_{|s|}(\tilde{\varepsilon})(q^s))^s$$

where $\alpha^s(i) \stackrel{X_i}{=} \alpha(i)(s * [\alpha^s](i))$ and $q^s(\alpha) = q(\alpha^s)$. Unfolding the definitions

$$\begin{aligned}
\text{IPS}_s(\varepsilon)(q)(i) &\stackrel{X_{|s|+i}}{=} (\text{ips}_{|s|}(\tilde{\varepsilon})(q^s))^s(i) \\
&= \text{ips}_{|s|}(\tilde{\varepsilon})(q^s)(i)(s * t) \\
&\stackrel{\text{D4.4}}{=} \tilde{\varepsilon}_{|s|+i}(\lambda f.(q^s)_{u*f}(\text{ips}_{|s|+i+1}(\tilde{\varepsilon})((q^s)_{u*f}))) (s * t) \\
&= \varepsilon_{s*t}(\lambda x.(q^s)_{u*\lambda r.x}(\text{ips}_{|s|+i+1}(\tilde{\varepsilon})((q^s)_{u*\lambda r.x}))) \\
&\stackrel{(i,ii)}{=} \varepsilon_{s*t}(\lambda x.q_{t*x}(\text{ips}_{|s|+i+1}(\tilde{\varepsilon})((q^s)_{u*\lambda r.x}))^{s*t*x}) \\
&\stackrel{(+)}{=} \varepsilon_{s*t}(\lambda x.q_{t*x}(\text{ips}_{|s|+i+1}(\tilde{\varepsilon})((q_{t*x})^{s*t*x}))^{s*t*x}) \\
&= \varepsilon_{s*t}(\lambda x.q_{t*x}(\text{IPS}_{s*t*x}(\varepsilon)(q_{t*x}))) \\
&= \varepsilon_{s*t}(\lambda x.\overline{\text{IPS}_{s*t*x}(\varepsilon)}(q_{t*x}))
\end{aligned}$$

where $t = [(\text{ips}_{|s|}(\tilde{\varepsilon})(q^s))^s](i)$ and $u = [\text{ips}_{|s|}(\tilde{\varepsilon})(q^s)](i)$. We used that

$$(i) \quad (t * \alpha)^s = t^s * (\alpha)^{s*t^s}, \text{ for all } t, s, \alpha,$$

(ii) $u^s = t$, for t, s and u as above.

Property (+) is again similarly proven as in Propositions 3.4, 3.13 and 3.15. \dashv

4.3. Relation to modified bar recursion. The proof that ips interprets full classical analysis, via modified realizability, is very similar to the one given in [4, 5] that *modified bar recursion* MBR interprets full classical analysis. In this section we show how MBR corresponds directly to the infinite iteration of a different form of binary product of selection functions. We also show (Sections 5.2 and 5.3) that this different product when iterated leads to a form of bar recursion (MBR) which is nevertheless T -equivalent to IPS.

DEFINITION 4.6. *Given a function $\varepsilon \in (X \rightarrow R) \rightarrow X \times Y$ and a selection function $\delta \in JY$ define a selection function $\varepsilon \tilde{\otimes} \delta \in J(X \times Y)$ as*

$$(\varepsilon \tilde{\otimes} \delta)(p) = \varepsilon(\lambda x.p(x, b(x)))$$

where $b(x) \stackrel{Y}{=} \delta(\lambda y.p(x, y))$. We shall also consider a dependent version $\tilde{\otimes}_d$ of the product where $\delta: X \rightarrow JY$ and $b(x) = \delta(x, \lambda y.p(x, y))$.

The above construction shows how a mapping of type $(X \rightarrow R) \rightarrow X \times Y$ can be extended to a selection function on the product space, given a simpler selection function on Y . We shall use this when $X = X_k$ and $Y = \Pi_i X_{i+k+1}$, so that we obtain a selection function in $J(\Pi_i X_{i+k})$.

DEFINITION 4.7 (mbr). *Define the iterated skewed product mbr as*

$$\text{mbr}_k(\varepsilon) \stackrel{J(\Pi_i X_{k+i})}{=} \varepsilon_k \tilde{\otimes} \text{mbr}_{k+1}(\varepsilon),$$

where $\varepsilon: \Pi_k((X_k \rightarrow R) \rightarrow \Pi_i X_{k+i})$. We name this mbr because we will show this is essentially modified bar recursion as defined in [4, 5].

We think of ε as a matrix of *skewed selection functions*. The idea is that sometimes a witness for X_k is automatically a witness for all types X_i for $i \geq k$. In such cases, a selection function $\varepsilon_n: (X_k \rightarrow R) \rightarrow X_k$ gives rise to a skewed selection function $\varepsilon: (X_k \rightarrow R) \rightarrow \Pi_{i \geq k} X_i$, so that the more intricate product of selection functions (Definition 2.5) can be replaced by the simpler product given

in Definition 4.6. As with IPS and EPS (Sections 3.2 and 4.2), we now show that also the *simple* iterated skewed product mbr (cf. Section 4.3) is T -equivalent to its *dependent* variant MBR.

THEOREM 4.8 (HA $^\omega$). $\text{mbr} \geq_T \text{MBR}$.

PROOF. For any given $\varepsilon: \Pi_j(\Pi_{i<j}X_i \rightarrow (X_j \rightarrow R) \rightarrow \Pi_k X_{j+k})$, let

$$A_j \equiv \mathbb{B} \times (\Pi_{i<j}X_i \rightarrow \Pi_k X_{j+k}),$$

and define

$$\tilde{\varepsilon}: \Pi_j((A_j \rightarrow R) \rightarrow \Pi_k A_{j+k})$$

by

$$\tilde{\varepsilon}_j(P^{A_j \rightarrow R})(k) \stackrel{A_{j+k}}{=} \begin{cases} \langle \text{tt}, \lambda t^{\Pi_{i<j+k}X_i} . \varepsilon_t(\lambda x^{X_j} . P(\hat{x})) \rangle & \text{if } k = 0, \\ \langle \text{ff}, \mathbf{0}^{A_{j+k}} \rangle & \text{if } k > 0, \end{cases}$$

where $\hat{x} = \langle \text{tt}, \lambda s^{\Pi_{i<j}X_i} . \langle x^{X_j}, \mathbf{0}^{X_{j+1}}, \mathbf{0}^{X_{j+2}}, \dots \rangle \rangle$. The simple skewed product of selection functions gives $\text{mbr}_0(\tilde{\varepsilon}): J(\Pi_i A_i)$. MBR can then be defined from mbr as

$$\text{MBR}_s(\varepsilon)(q^{\Pi_i X_{|s|+i} \rightarrow R}) \stackrel{\Pi_i X_{|s|+i}}{=} (\text{mbr}_{|s|}(\tilde{\varepsilon})(q^s))^s.$$

Here, for $\alpha: \Pi_i A_{|s|+i}$,

$$q^s(\alpha) = q(\alpha^s)$$

and

$$\alpha^s(i) \stackrel{\Pi_i X_{|s|+i}}{=} \begin{cases} f(s * [\alpha^s](i))(0) & \text{if } \alpha(i) = \langle \text{tt}, f^{\Pi_{j<|s|+i}X_i \rightarrow \Pi_k X_{|s|+i+k}} \rangle, \\ g(s * [\alpha^s](n))(i - n) & \text{otherwise,} \end{cases}$$

where n is the last position before i where $\alpha(n)$ is of the form $\langle \text{tt}, g \rangle$. Unfolding the definitions we have

$$\begin{aligned} \text{MBR}_s(\varepsilon)(q) &\stackrel{\Pi_i X_{|s|+i}}{=} (\text{mbr}_{|s|}(\tilde{\varepsilon})(q^s))^s \\ &= (\tilde{\varepsilon}_{|s|}(\lambda f^{A_{|s|}} . (q^s)_f(\text{mbr}_{|s|+1}(\tilde{\varepsilon})((q^s)_f))))^s \\ &= \varepsilon_s(\lambda x^{X_{|s|}} . (q^s)_{\hat{x}}(\text{mbr}_{|s|+1}(\tilde{\varepsilon})((q^s)_{\hat{x}}))) \\ &= \varepsilon_s(\lambda x^{X_{|s|}} . q_x((\text{mbr}_{|s*x|}(\tilde{\varepsilon})((q_x)^{s*x}))^{s*x})) \\ &= \varepsilon_s(\lambda x^{X_{|s|}} . q_x(\text{MBR}_{s*x}(\varepsilon)(q_x))), \end{aligned}$$

using that $(q^s)_{\hat{x}}(\alpha) = q_x(\alpha^{s*x}) = (q_x)^{s*x}(\alpha)$. \dashv

We now show that a slight generalisation of modified bar recursion [4, 5] is T -equivalent to the iterated product of skewed selection functions.

THEOREM 4.9 (HA $^\omega$ + C-BI). Define MBR' as

$$\text{MBR}'_s(\varepsilon)(q) \stackrel{\Pi_i X_i}{=} s * \varepsilon_s(\lambda x^{X_{|s|}} . q(\text{MBR}'_{s*x}(\varepsilon)(q)))$$

where $q: \Pi_i X_i \rightarrow R$ and $\varepsilon: \Pi_k(\Pi_{i<k}X_i \times (X_k \rightarrow R) \rightarrow \Pi_i X_{k+i})$. Then mbr and MBR' are T -equivalent.

PROOF. MBR' is a generalisation of modified bar recursion to sequence types. If all $X_i = X$ we have precisely modified bar recursion, as defined in [4, 5]. As just shown, mbr is T -equivalent to its dependent version MBR . Therefore, for one direction, let $q: \Pi_i X_i \rightarrow R$ and $s: \Pi_{i < n} X_i$ and define

$$\text{MBR}'_s(\varepsilon)(q) \stackrel{\Pi_i X_i}{=} s * \text{MBR}_s(\varepsilon)(q_s).$$

Unfolding definitions we have

$$\begin{aligned} \text{MBR}'_s(\varepsilon)(q) &= s * \text{MBR}_s(\varepsilon)(q_s) \\ &= s * ((\varepsilon_s \otimes_d \lambda x. \text{MBR}_{s*x}(\varepsilon))(q_s)) \\ &= s * \varepsilon_s(\lambda x^{X|s|.} . q_{s*x}(\text{MBR}_{s*x}(\varepsilon)(q_{s*x}))) \\ &= s * \varepsilon_s(\lambda x^{X|s|.} . q(s * x * \text{MBR}_{s*x}(\varepsilon)(q_{s*x}))) \\ &= s * \varepsilon_s(\lambda x^{X|s|.} . q(\text{MBR}'_{s*x}(\varepsilon)(q))). \end{aligned}$$

For the other direction, let $q: \Pi_i X_{|s|+i} \rightarrow R$. Define

$$\text{MBR}_s(\varepsilon)(q) \stackrel{\Pi_i X_{|s|+i}}{=} \text{MBR}'_{\langle \rangle}(\lambda t. \varepsilon_{s*t})(q).$$

We then have

$$\begin{aligned} \text{MBR}_s(\varepsilon)(q) &= \text{MBR}'_{\langle \rangle}(\lambda t. \varepsilon_{s*t})(q) \\ &= \varepsilon_s(\lambda x^{X|s|.} . q(\text{MBR}'_x(\lambda t. \varepsilon_{s*t})(q))) \\ &\stackrel{(*)}{=} \varepsilon_s(\lambda x^{X|s|.} . q_x(\text{MBR}'_{\langle \rangle}(\lambda t. \varepsilon_{s*x*t})(q_x))) \\ &= \varepsilon_s(\lambda x^{X|s|.} . q_x(\text{MBR}_{s*x}(\varepsilon)(q_x))), \end{aligned}$$

where $(*) \text{MBR}'_{r*r'}(\lambda t. \varepsilon_{s*t})(q) = r * \text{MBR}'_{r'}(\lambda t. \varepsilon_{s*r*t})(q_r)$ can be proven by bar induction on the sequence r' , assuming continuity of q . \dashv

COROLLARY 4.10. *Gandy's functional Γ is T -equivalent to MBR with $X_i = \mathbb{N}$ for all $i \in \mathbb{N}$.*

PROOF. It has been shown in [5] that the Γ functional is T -equivalent to MBR of lowest type. It remains to observe that the equivalence of Theorem 4.9 respects the types. \dashv

QUESTION 4.11. *It should be mentioned that in [2] yet another form of bar recursion is used for the interpretation of countable choice (although they also use modified bar recursion when dealing with dependent choice). We refer to this different bar recursion as the bbc functional. It completely open how the bbc functional fits into the picture in Figure 1. For more information on the bbc functional see [3].*

§5. Further Inter-definability Results. Figure 1 gives a diagrammatic picture of all known results concerning T -definability between the different forms of bar recursion, and products of selection functions and quantifiers. Some of the results are known, and some have already been proved above. In this section we prove the remaining results which are depicted in this diagram.

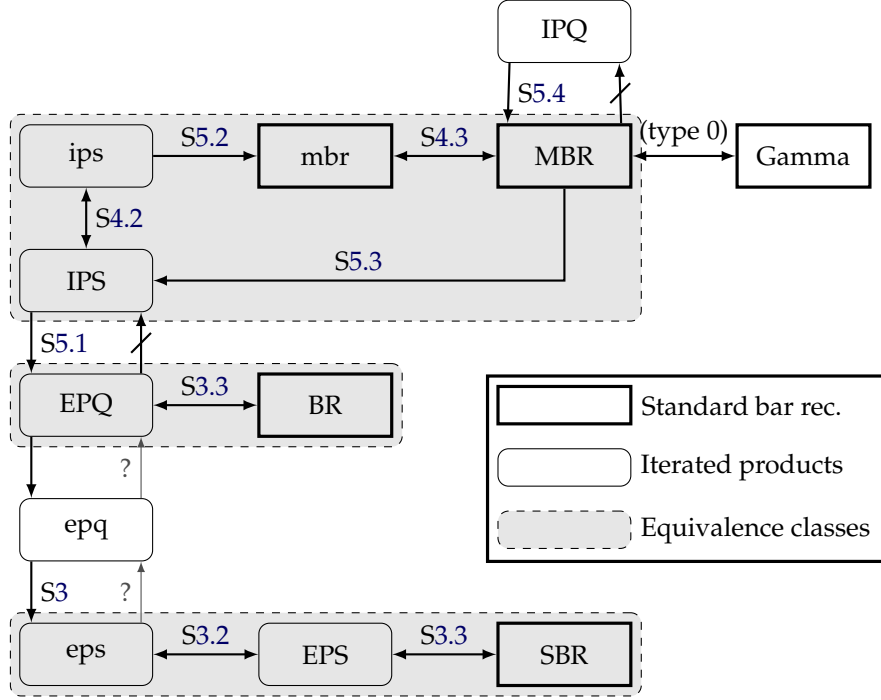


FIGURE 1. Diagram of inter-definability results

5.1. $IPS \geq EPQ$. It has been shown in [5] that BR is T -definable from modified bar recursion. Here we simplify that construction and use it to show that EPQ is T -definable from IPS. First we prove that *Spector's search functional* is definable in Gödel's system T .

LEMMA 5.1 (HA^ω). *The Spector's search functional*

$$\mu_{sc}(\omega)(\alpha) = \text{least } n(\omega(\bar{\alpha}, \bar{n}) < n)$$

is T -definable.

PROOF. We show how this unbounded search can be turned into a bounded search. Abbreviate $A_n(\omega, \alpha) = (\omega(\bar{\alpha}, \bar{n}) < n)$. Consider the following construction, given $\alpha: \Pi_i X_i$ define α^ω as

$$\alpha^\omega(i) = \begin{cases} \mathbf{0} & \text{if } \exists k \leq i + 1 A_k(\omega, \alpha) \\ \alpha(i) & \text{otherwise.} \end{cases}$$

Let n is the least number such that $A_n(\omega, \alpha)$ holds. Then it is easy to see that $\alpha^\omega = \alpha, n - 1$. Because n is the least, we must have that $\omega(\alpha^\omega) \geq n - 1$, and hence $n \leq \omega(\alpha^\omega) + 1$. Therefore, $\omega(\alpha^\omega) + 1$ serves as an upper bound on the search μ_{sc} . \dashv

It had been normally assumed that Spector's search functional was not definable in system T , although it is clearly definable in T extended with Spector's bar recursion as shown in [15].

REMARK 5.2. *In particular we have that bar recursion BR as formulated in Section 3.3 is T-equivalent to the version where we slightly change the stopping condition to guarantee monotonicity, as used in [12],*

$$\text{BR}'_s(\omega)(\phi)(q) \stackrel{R}{=} \begin{cases} q_t(\mathbf{0}) & \text{if } \exists t \preceq s (\omega_t(\mathbf{0}) < |t|) \\ \phi_s(\lambda x^{X_{|s|}}. \text{BR}'_{s*x}(\omega)(\phi)(q)) & \text{otherwise,} \end{cases}$$

where $t \preceq$ is the shortest sequence such that $(\omega_t(\mathbf{0}) < |t|)$. Notice that if s is such that $\omega_s(\mathbf{0}) < |s|$ then $g(\hat{t})$, with t the least prefix of s also satisfying the condition, can be computed as $g(\hat{t}) = \tilde{g}(\hat{s})$ where $\tilde{g}(\alpha) = g(\overline{\alpha}, \bar{n})$ and $n = \text{least } n (\omega(\overline{\alpha}, \bar{n}) < n)$, which is a T-definable construction by Lemma 5.1. The same holds for SBR.

THEOREM 5.3 (HA $^\omega$ + C-BI). $\text{IPS} \geq_T \text{EPQ}$.

PROOF. We use a generalisation of the search operator above, namely

$$\mu_{\text{sc}}^k(\omega)(\alpha) = \text{least } n (\omega(\overline{\alpha}, \bar{n}) < k + n),$$

which is clearly definable in μ_{sc} . Let $\psi_k: \Pi_{i < k} X_i \rightarrow KX_k$ be a given indexed family of quantifiers. Let also $X \uplus Y$ denote the sum of types X and Y , which can be implemented as $\mathbb{B} \times X \times Y$, since we assume all types are inhabited. Let $\text{inj}_X: X \rightarrow X \uplus Y$ and $\text{inj}_Y: Y \rightarrow X \uplus Y$ be the standard injections. We first turn each family of quantifiers ψ_k into a family of selection functions $\tilde{\psi}_k: \Pi_{i < k} (X_i \uplus R) \rightarrow J(X_k \uplus R)$ as

$$\tilde{\psi}_k(s^{\Pi_{i < k} (X_i \uplus R)}, F^{(X_k \uplus R) \rightarrow R}) \stackrel{X_k \uplus R}{=} \text{inj}_R(\psi_k(\check{s}, \lambda x^{X_k}. F(\text{inj}_{X_k} x)))$$

where

$$(\check{s})_i \stackrel{X_i}{=} \begin{cases} x_i & \text{if } s_i = \text{inj}_{X_i}(x_i) \\ \mathbf{0}^{X_i} & \text{otherwise.} \end{cases}$$

We will also make use of the dual operation that given an $s: \Pi_{i < k} X_i$ injects this into $\tilde{s}: \Pi_{i < k} (X_i \uplus R)$. Finally, given $q: \Pi_i X_{k+i} \rightarrow R$ and $\omega: \Pi_i X_i \rightarrow \mathbb{N}$ define $q^{\omega, s}: \Pi_i (X_{k+i} \uplus R) \rightarrow R$ as

$$q^{\omega, s}(\alpha^{\Pi_i (X_{k+i} \uplus R)}) \stackrel{R}{=} \begin{cases} q([\check{\alpha}](N) * \mathbf{0}) & \text{if } \forall i < N (\alpha(i) \in X_{k+i}) \\ a & \text{otherwise,} \end{cases}$$

where $N = \mu_{\text{sc}}^{|\check{\alpha}|}(\omega_s)(\check{\alpha})$ and $\alpha(\mu i < N (\alpha(i) \in R)) = \text{inj}_R(a)$. Intuitively, when $q^{\omega, s}$ reads an input sequence, it uses Spector's condition functional on ω to compute the point N where Spector's condition is satisfied in $\check{\alpha}$. If all values in α up to that point are X_i values, then we apply q . Otherwise, some value is encoding the return of the computation R , and the first such value is then returned. We claim that

$$\text{EPQ}_s(\omega)(\psi)(q^{\Pi_i X_{|s|+i} \rightarrow R}) \stackrel{R}{=} q^{\omega, s}(\text{IPS}_{\tilde{s}}(\tilde{\psi})(q^{\omega, s}))$$

does the job. Let us unfold the definitions. Assume first that $\omega_s(\mathbf{0}) < |s|$. Then

$$\begin{aligned} \text{EPQ}_s(\omega)(\psi)(q) &\stackrel{R}{=} q^{\omega, s}(\text{IPS}_{\tilde{s}}(\tilde{\psi})(q^{\omega, s})) \\ &= q(\mathbf{0}). \end{aligned}$$

On the other hand, if $\omega_s(\mathbf{0}) \geq |s|$, we have

$$\begin{aligned}
\text{EPQ}_s(\omega)(\psi)(q) &\stackrel{R}{=} q^{\omega,s}(\text{IPS}_{\tilde{s}}(\tilde{\psi})(q^{\omega,s})) \\
&= \psi_s(\lambda x.(q^{\omega,s})_{\text{inj}_{X_{|s|}}(x)}(\text{IPS}_{\tilde{s} * \text{inj}_{X_{|s|}}(x)}(\tilde{\psi})((q^{\omega,s})_{\text{inj}_{X_{|s|}}(x)}))) \\
&\stackrel{(+)}{=} \psi_s(\lambda x.(q_x)^{\omega,s*x}(\text{IPS}_{\tilde{s} * \tilde{x}}(\tilde{\psi})((q_x)^{\omega,s*x}))) \\
&= \psi_s(\lambda x.\text{EPQ}_{s*x}(\omega)(\psi)(q_x)) \\
&= (\psi_s \otimes_d \lambda x.\text{EPQ}_{s*x}(\omega)(\psi))(q),
\end{aligned}$$

using that $(q^{\omega,s})_{\text{inj}_{X_{|s|}}(x)} = (q_x)^{\omega,s*x}$ and that $\tilde{s} * \text{inj}_{X_{|s|}}(x) = \tilde{s} * \tilde{x}$. We use C-BI to show step (+), as in Propositions 3.4, 3.13, 3.15 and 4.5. \dashv

5.2. $\text{ips} \geq \text{mbr}$. The main difference between MBR and IPS is that in the recursive call for the n -th value the recursion IPS uses potentially all previous values below n . Moreover, in MBR the selection function ε_i updates the sequence s at point $|s|$ whereas IPS updates it at position i . Nevertheless, we show the two forms of bar recursion are T -equivalent. In this section we show that MBR is T -definable from IPS. The converse will be shown in the following section.

THEOREM 5.4 (HA $^\omega$). $\text{ips} \geq_T \text{mbr}$.

PROOF. A family of mappings $\varepsilon_i: (X_i \rightarrow R) \rightarrow \Pi_j X_{i+j}$ can be turned into a family of selection functions $\tilde{\varepsilon}_i: J(\Pi_j X'_{i+j})$ as

$$\tilde{\varepsilon}_i(f^{\Pi_j X'_{i+j} \rightarrow R}) \stackrel{\Pi_j X'_{i+j}}{=} \lambda j. \langle \text{ff}, \varepsilon_i(\lambda x^{X_i}. f(\hat{x}))(j) \rangle,$$

where

$$\hat{x}(j) = \begin{cases} \langle \text{tt}, x^{X_i} \rangle & \text{if } j = 0 \\ \langle \text{tt}, \mathbf{0}^{X_{i+j}} \rangle & \text{if } j > 0. \end{cases}$$

Also, $q: \Pi_j X_{i+j} \rightarrow R$ can be turned into $\tilde{q}: \Pi_j \Pi_k X'_{i+j+k} \rightarrow R$ as $\tilde{q}(\alpha) = q(\tilde{\alpha})$ where

$$\tilde{\alpha}(j) \stackrel{X_{i+j}}{=} \begin{cases} (\alpha(j)(0))_1 & \text{if } \forall k < j ((\alpha(k)(0))_0 = \text{tt}) \\ (\alpha(k)(j-k))_1 & \text{otherwise,} \end{cases}$$

where $k = \mu k < j (\alpha(k)(0))_0 = \text{ff}$. Let $\varepsilon: \Pi_i ((X_i \rightarrow R) \rightarrow \Pi_j X_{i+j})$ and $q: \Pi_j X_{i+j} \rightarrow R$. We claim that mbr can be defined as

$$\text{mbr}_i(\varepsilon)(q) \stackrel{\Pi_j X_{i+j}}{=} ((\text{ips}_i(\tilde{\varepsilon})(\tilde{q}))^{\Pi_j \Pi_k X'_{i+j+k}}(0))_1.$$

Unfolding definitions

$$\begin{aligned}
\text{mbr}_i(\varepsilon)(q) &\stackrel{\Pi_j X_{i+j}}{=} (\text{ips}_i(\tilde{\varepsilon})(\tilde{q})(0))^1 \\
&= (\tilde{\varepsilon}_i(\lambda x^{\Pi_{j \geq i} X_j^i} . \tilde{q}_\alpha(\text{ips}_{i+1}(\tilde{\varepsilon})(\tilde{q}_\alpha))))^1 \\
&= \varepsilon_i(\lambda x^{X_i} . \tilde{q}_{\tilde{x}}(\text{ips}_{i+1}(\tilde{\varepsilon})(\tilde{q}_{\tilde{x}}))) \\
&= \varepsilon_i(\lambda x^{X_i} . \tilde{q}_x(\text{ips}_{i+1}(\tilde{\varepsilon})(\tilde{q}_x))) \\
&= \varepsilon_i(\lambda x^{X_i} . q_x((\text{ips}_{i+1}(\tilde{\varepsilon})(\tilde{q}_x)(0))^1)) \\
&= \varepsilon_i(\lambda x^{X_i} . q_x(\text{mbr}_{i+1}(\varepsilon)(q_x))) \\
&= (\varepsilon_i \otimes \text{mbr}_{i+1}(\varepsilon))(q),
\end{aligned}$$

using that $\tilde{q}_{\tilde{x}} = \tilde{q}_x$. \dashv

5.3. MBR \geq IPS. We now show that the implicitly controlled product of selection functions IPS is in fact T -equivalent to MBR. The proof that IPS defines MBR in Gödel's system T , however, will make use of the assumption that finite sequences of X_i 's can be coded as single elements. This is true in the particular case when all X_i are the same (as it is usually considered), but might not be the case in the full general case where arbitrary dependent types are permitted. In such a general case it might still be that MBR is weaker than IPS over system T . We start by considering equivalent variants of MBR.

LEMMA 5.5 ([4], lemma 2). *MBR is T -equivalent to*

$$(4) \quad \text{MBR}_s^0(\varepsilon)(q)(i) \stackrel{X_i}{=} \begin{cases} s_i & \text{if } i < |s| \\ \varepsilon_s(\lambda r \lambda x^{X_j} . q(\text{MBR}_{s**r*x}^0(\varepsilon)(q)))(i) & \text{otherwise,} \end{cases}$$

where $r: \Pi_{|s| \leq k < j} X_k$.

PROOF. This is a variant of MBR where the bar recursion can make use of any value bar recursively computed, and not just the immediate children $s * x$ of the node s . MBR^0 is clearly more general than MBR. The fact that MBR T -defines MBR^0 is very similar to the one given in ([4], lemma 2) and hence omitted here. We only note that the proof of equivalence can be carried out in HA^ω . \dashv

The following T -equivalent formulation of MBR allows the selection functions ε_s to access the course-of-value results computed so far. Using an analogy with standard primitive recursion, this is like moving from iteration to recursion. It is well-know that primitive recursion is definable from simple iteration. We show that the analogous situation holds for MBR as well.

THEOREM 5.6 (HA^ω). *MBR^0 is T -equivalent to*

$$(5) \quad \text{MBR}_s^1(\varepsilon)(q)(i) \stackrel{X_i}{=} \begin{cases} s_i & \text{if } i < |s| \\ \varepsilon_s(t_{s,i}, \lambda r \lambda x^{X_j} . q(\text{MBR}_{s**r*x}^1(\varepsilon)(q)))(i) & \text{otherwise,} \end{cases}$$

where $r: \Pi_{|s| \leq k < j} X_k$ and $t_{s,i} = \text{MBR}_s^1(\varepsilon)(q)[|s|, i - 1]$.

PROOF. This can be viewed as a form of modified bar recursion combined with course-of-values recursion. In order to define the point i of the infinite sequence MBR_s^1 we are allowed to use MBR_s^1 up to point $i - 1$. MBR^1 is clearly

more general than MBR^0 . For the other direction, consider the following construction

$$\hat{\varepsilon}_s(f)(i) = \varepsilon_s(t_{s,i}, f)(i)$$

where $t_{s,n} = h_s(0) * \dots * h_s(n - |s|)$, and $h_s(i)$ is calculated by course-of-value recursion as

$$h_s(i) = \varepsilon_s([h_s](i), f)(|s| + i).$$

Define MBR^1 from MBR^0 as $\text{MBR}_s^1(\varepsilon)(q) = \text{MBR}_s^0(\hat{\varepsilon})(q)$. We show that MBR^1 satisfies equation (5). Let $i \geq |s|$ (if $i < |s|$ the result is trivial), we have

$$\begin{aligned} \text{MBR}_s^1(\varepsilon)(q)(i) &= \text{MBR}_s^0(\hat{\varepsilon})(q)(i) \\ &= \hat{\varepsilon}_s(\lambda r, x. \text{MBR}_{s*r*x}^0(\hat{\varepsilon})(q))(i) \\ &= \hat{\varepsilon}_s(\lambda r, x. \text{MBR}_{s*r*x}^1(\varepsilon)(q))(i) \\ &= \varepsilon_s(t_{s,i}, \lambda r, x. \text{MBR}_{s*r*x}^1(\varepsilon)(q))(i), \end{aligned}$$

where $t_{s,i} = \text{MBR}_s^1(\varepsilon)(q)[|s|, i - 1]$. −

As a straightforward consequence of the above lemmas we obtain that MBR is T -definable in IPS .

COROLLARY 5.7 (HA^ω). *IPS is T-definable from MBR.*

PROOF. Let $\text{IPS}_s(\varepsilon)(q) = \text{MBR}_s^1(\tilde{\varepsilon})(q)$, where $\tilde{\varepsilon}_s(t, f) = \varepsilon_s(ft)$. −

5.4. $\text{IPQ} \geq \text{MBR}$. Finally, we conclude with the observation that MBR is also T -defined from IPQ . Given a family of functions

$$\varepsilon_s: (X_{|s|} \rightarrow R) \rightarrow \Pi_i X_{|s|+i}$$

where $s: \Pi_{j < |s|} X_j$, and a predicate $q: \Pi_i X_i \rightarrow R$, define the following family of quantifiers as

$$\phi_s^{\varepsilon, q}(p^{X_{|s|} \rightarrow R}) \stackrel{R}{=} q_s(\varepsilon_s(p)).$$

Then MBR can be defined from IPQ as

$$\text{MBR}_s(\varepsilon)(q) \stackrel{\Pi_i X_{|s|+i}}{=} \varepsilon_s(\lambda x^{X_{|s|}}. \text{IPQ}_{s*x}(\phi_s^{\varepsilon, q^{|s|}})(q_x)),$$

where $q^n(\alpha^{\Pi_i X_i}) = q(\lambda i. \alpha(n + i))$. We have

$$\begin{aligned} \text{MBR}_s(\varepsilon)(q) &= \varepsilon_s(\lambda x^{X_{|s|}}. \text{IPQ}_{s*x}(\phi_s^{\varepsilon, q^{|s|}})(q_x)) \\ &= \varepsilon_s(\lambda x^{X_{|s|}}. \phi_{s*x}^{\varepsilon, q^{|s|}}(\lambda y^{X_{|s|+1}}. \text{IPQ}_{s*x*y}(\phi_s^{\varepsilon, q^{|s|}})(q_{x*y}))) \\ &= \varepsilon_s(\lambda x^{X_{|s|}}. (q^{|s|})_{s*x}(\varepsilon_{s*x}(\lambda y^{X_{|s|+1}}. \text{IPQ}_{s*x*y}(\phi_s^{\varepsilon, q^{|s|}})(q_{x*y})))) \\ &= \varepsilon_s(\lambda x^{X_{|s|}}. q_x(\varepsilon_{s*x}(\lambda y^{X_{|s|+1}}. \text{IPQ}_{s*x*y}(\phi_s^{\varepsilon, q^{|s|}})(q_{x*y})))) \\ &= \varepsilon_s(\lambda x^{X_{|s|}}. q_x(\varepsilon_{s*x}(\lambda y^{X_{|s*x|}}. \text{IPQ}_{s*x*y}(\phi_s^{\varepsilon, (q_x)^{|s*x|}})(q_{x*y})))) \\ &= \varepsilon_s(\lambda x^{X_{|s|}}. q_x(\text{MBR}_{s*x}(\varepsilon)(q_x))) \\ &= (\varepsilon_s \tilde{\otimes}_d \lambda x^{X_{|s|}}. \text{MBR}_{s*x}(\varepsilon))(q). \end{aligned}$$

Note, however, that IPQ is not defined in general, but MBR is, assuming continuity. The reason is that under the continuity assumption $\phi_{\varepsilon, p}$ eventually (for long enough s) becomes constant.

§6. **Non-definability results.** It is well known that in the model \mathcal{C} of continuous functionals there is a functional $\text{fan}: (\mathbb{B}^\omega \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ such that for every $f: \mathbb{B}^\omega \rightarrow \mathbb{N}$ in the model,

$$\forall i < \text{fan}(f) (\alpha_i = \beta_i) \implies f(\alpha) = f(\beta).$$

That is, $\text{fan}(f)$ is a modulus of uniform continuity of f . We may assume without loss of generality, because we can perform bounded search in system T , that $\text{fan}(f)$ is the smallest modulus of uniform continuity, which makes the fan functional unique. It is also well known that the fan functional of the model \mathcal{C} is not S1–S9 definable. From the fan functional one can easily T -define a functional

$$\text{forall}: (\mathbb{B}^\omega \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$$

such that, for every $p: \mathbb{B}^\omega \rightarrow \mathbb{B}$ in the model,

$$\text{forall}(p) = 1 \iff \forall \alpha (p(\alpha) = 1).$$

THEOREM 6.1. *The forall functional of model \mathcal{C} is not T -definable.*

Before proving Theorem 6.1, we formulate two consequences. The first consequence gives further non-definability results.

COROLLARY 6.2. *The fan functional and the countable product of selection functions of model \mathcal{C} are not T -definable, and there is no T -definable selection function for the Cantor space \mathbb{B}^ω in \mathcal{C} .*

PROOF. Because the fan functional T -defines forall, because a selection function for the Cantor space also T -defines forall, and because the countable product of selection functions T -defines a selection function for the Cantor space, as shown in [8]. \dashv

The second consequence gives a model independent view of the non definability result formulated in the theorem.

COROLLARY 6.3. *HA^ω does not prove the formula*

$$\exists F^{(\mathbb{B}^\omega \rightarrow \mathbb{B}) \rightarrow \mathbb{B}} \forall p^{\mathbb{B}^\omega \rightarrow \mathbb{B}} (F(p) = 1 \iff \forall \alpha^{\mathbb{B}^\omega} (p(\alpha) = 1)).$$

PROOF. From such a proof one would be able to extract a system T term whose interpretation in the model \mathcal{C} is a functional $\text{forall}: (\mathbb{B}^\omega \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$ such that for every functional $p: \mathbb{B}^\omega \rightarrow \mathbb{B}$ in model \mathcal{C} , one has that $\text{forall}(p) = 1$ iff $p(\alpha) = 1$ for all $\alpha \in \mathbb{B}^\omega$. This would contradict Theorem 6.1, which concludes the proof. \dashv

Notice that, by the previous development, we also have that HA^ω does not prove the evident, model independent, formulations of the existence of a fan functional or of a selection function for the Cantor space.

It is also well-known, on the other hand, that in the model HEO of hereditarily effective operations [19] there are no forall or fan functionals satisfying the above specifications. However, in the model \mathcal{M} of majorizable functionals [7], there is a forall functional but there is no fan functional (cf. Lemma 6.11 below). This is exploited in the proof of Theorem 6.7 below, which shows that the non definability of the forall functional is a stronger result than the non definability of the fan functional.

To prove Theorem 6.1, we consider the Scott model of system T [17]. We consider the notion of \mathcal{C} -totality in the Scott model, defined by induction on types: At ground types, an element is called \mathcal{C} -total iff it is distinct from bottom. At product types, an element is \mathcal{C} -total iff every coordinate is \mathcal{C} -total. At function types, an element is \mathcal{C} -total iff it maps \mathcal{C} -total elements to \mathcal{C} -total elements.

The Scott model admits a notion of computability. One defines enumerations of the compact (or finite) elements, and an element is said to be computable iff its compact approximants form an r.e. set. We define the HEO-*total* elements of the sub-model of computable elements by induction on types as follows: At ground types, all elements are computable, and an element is HEO-total iff it is \mathcal{C} -total. At product types, a computable element is HEO-total iff every coordinate is HEO-total. At function types, a computable element is HEO-total iff it maps HEO-total elements to HEO-total elements.

LEMMA 6.4. *Every T -definable element of the Scott model is computable, \mathcal{C} -total and HEO-total.*

This is well known with an easy proof by induction on the formation of system T terms. The following is also well known.

LEMMA 6.5. *There is a non-compact, HEO-total element $p: \mathbb{B}_\perp^\omega \rightarrow \mathbb{B}_\perp$ of the Scott model with $p(\alpha) = 1$ for every HEO-total $\alpha \in \mathbb{B}_\perp^\omega$.*

PROOF. Consider a Kleene singular tree, that is, a decidable, binarily branching, infinite tree with no computable infinite paths. Define p by the following algorithm: $p(\alpha) = 1$ if there is n such that $\alpha_i \neq \perp$ for every $i < n$ and such that the finite prefix $[\alpha](n)$ is not in the tree, and otherwise $p(\alpha) = \perp$. If α is HEO-total then $p(\alpha) = 1$, because otherwise every prefix of α would be in the tree and α would be a computable infinite path. Hence p is HEO-total, because it is computable. Now, $\alpha \in \mathbb{B}_\perp^\omega$ is compact iff $\alpha_i = \perp$ for all but finitely many i , and p is compact iff it can be written as a finite join $\bigsqcup_i s_i \mapsto b_i$ with $s_i \in \mathbb{B}_\perp^\omega$ compact and $b_i \in \mathbb{B}$, where the step function $s_i \mapsto b_i$ is the smallest continuous function that maps s_i to b_i . Hence if p were compact, there would be finitely many minimal words $s \in \mathbb{B}^*$ not in the tree, and hence because the tree is infinite, it would have the full binary tree as a subtree, and hence a computable infinite path, leading to a contradiction. \dashv

Call this a *Kleene singular predicate*. Notice that such a predicate is not \mathcal{C} -total, because a \mathcal{C} -total predicate on the Cantor space is necessarily compact. (This shows that the Kleene singular predicate is not T -definable, in view of Lemma 6.4, but we do not rely on this fact.) As shown by Ershov, the model \mathcal{C} can be obtained from the Scott model by quotienting the \mathcal{C} -total elements by a suitable equivalence relation, where members of the equivalence classes are called *representatives* of the element of \mathcal{C} that they correspond to [17]. (Ershov also showed that the model HEO can be obtained from the Scott model by suitably quotienting the HEO-total elements, and this is the reason for our terminology HEO-*total*, but we do not use this fact.)

LEMMA 6.6. *Every representative $A: (\mathbb{B}_\perp^\omega \rightarrow \mathbb{B}_\perp) \rightarrow \mathbb{B}_\perp$ of the forall functional of model \mathcal{C} fails to be HEO-total.*

PROOF. Assume that some representative A is HEO-total. Then $A(p)$ is defined for the Kleene singular predicate p , because p is HEO-total. Now, by continuity of A , there is a compact $p' \sqsubseteq p$ with $A(p') = A(p)$. We will define two predicates $p_0 \sqsupseteq p'$ and $p_1 \sqsupseteq p'$ such that $A(p_0) = 0$ and $A(p_1) = 1$. This gives a contradiction, because by monotonicity of A and the fact that $A(p') \neq \perp$, we get $A(p_0) = A(p') = A(p_1)$. We can take p_1 to be simply the constant function $\lambda x.1$. The construction of p_0 is more elaborate. Because p' is compact below p , it is $\bigsqcup\{s \mapsto 1 \mid s \in F_1\}$ for some finite set F_1 of compact elements of the domain \mathbb{B}_\perp^ω . Define the rank of a compact element $\alpha \in \mathbb{B}_\perp^\omega$ to be the first index n such that $\alpha_i = \perp$ for all $i \geq n$. Let m be the maximum of the ranks of the compact elements collected in F_1 , and F_0 be the compact elements of rank m which are inconsistent with all elements of F_1 . Then p' is consistent with the predicate $p'' = \bigsqcup\{s \mapsto 0 \mid s \in F_0\}$, and because F_0 is finite, the predicate $p_0 = p' \sqcup p''$ is compact. It is also \mathcal{C} -total, because for any \mathcal{C} -total $\alpha \in \mathbb{B}_\perp^\omega$, the first m elements of α determine a compact below α of rank m that is either above a member of F_1 or in F_0 , and hence mapped by p_0 to either 1 or 0. Because p is not \mathcal{C} -total, there is a \mathcal{C} -total $\alpha \in \mathbb{B}_\perp^\omega$ such that $p(\alpha) = \perp$. For such an α , the compact element $s \sqsubseteq \alpha$ of rank m is in F_0 , and hence $p_0(\alpha) = 0$. Therefore $A(p_0) = 0$ by specification of A , as required. \dashv

It follows from this and Lemma 6.4 that no representative of the forall functional can be T -definable, which concludes the proof of Theorem 6.1. There are \mathcal{C} -total, computable representatives of the forall functional, but they map the Kleene singular predicate to \perp , and hence they are not HEO-total because the Kleene singular predicate is HEO-total. At type $\mathbb{B}_\perp^\omega \rightarrow \mathbb{B}_\perp$, the notion of \mathcal{C} -totality is a much more restrictive condition than HEO-totality, and it is this that makes the existence of the functional forall: $(\mathbb{B}^\omega \rightarrow \mathbb{B}) \rightarrow \mathbb{B}$ in the model \mathcal{C} possible. (At higher types, the two notions of totality become incomparable.)

A selection function for the Cantor space in model \mathcal{C} is T -definable from the forall functional, as shown in [8] (the construction given in that reference does not mention system T explicitly, but clearly can be carried out in system T). On the other hand:

THEOREM 6.7. *The fan functional of model \mathcal{C} is not T -definable from the forall functional.*

COROLLARY 6.8. *HA^ω does not prove*

$$\exists F \text{IsForall}(F) \rightarrow \exists G \text{IsFan}(G),$$

where $\text{IsForall}(F)$ and $\text{IsFan}(G)$ are the evident, model independent, specifications of the forall and fan functionals expressed in HA^ω .

The proof is as that of Corollary 6.3. In order to prove Theorem 6.7, we simultaneously define, for elements x and y of model \mathcal{C} , binary and unary majorization relations, “ x majorizes y ” and “ y is majorizable” by induction on types. At ground types (natural numbers or booleans $0, 1$), we say that an element x majorizes an element y iff $x \geq y$, and any element is majorizable (by itself). At function types, we say that f majorizes g iff $f(x)$ majorizes both $g(y)$ and $f(y)$

whenever x majorizes y , for all majorizable x and y , and we say that g is majorizable iff it is majorized by some f . Countable products are treated as function types. At finite products, majorization and majorizability are defined coordinatewise. A similar construction appears in [7] where the full set-theoretic model is used as a starting point, instead of the model \mathcal{C} . Here we are adapting Bezem's notion of *strong* majorizability, rather than Howard's original notion of majorizability [14], as the former gives rise to more direct proofs.

LEMMA 6.9. *The majorizable elements of model \mathcal{C} form a submodel $\mathcal{C}_{\mathcal{M}}$ of system T .*

PROOF. It is enough to show that the S and K combinators, the recursion combinator, and the projections of product types are majorizable. This is a straightforward adaptation of the proof that \mathcal{M} is a model of system T . \dashv

COROLLARY 6.10. *If an element y in model \mathcal{C} is T -definable from a majorizable element x of model \mathcal{C} , then y is majorizable.*

PROOF. Let f be a T -definable function of model \mathcal{C} with $y = f(x)$. Then f is majorizable by Lemma 6.9, and, by definition of majorizability, y is majorizable because x is majorizable. \dashv

LEMMA 6.11. *(1) The forall functional of model \mathcal{C} is majorizable, but (2) the fan functional of model \mathcal{C} is not majorizable.*

PROOF. (1) The forall functional is majorized by the constant function $\lambda p.1$. (2) Assume the fan functional of \mathcal{C} is majorized by some $F: (\mathbb{B}^\omega \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$. Then $F(\lambda\alpha.1) \geq \text{fan } f$ for any f majorized, and hence bounded, by $\lambda\alpha.1$. This is a contradiction, because we can construct a continuous function $f: \mathbb{B}^\omega \rightarrow \mathbb{N}$ bounded by $\lambda\alpha.1$ with modulus of uniform continuity bigger than the number $F(\lambda\alpha.1)$, for example $f(\beta) = \beta_{F(\lambda\alpha.1)}$. \dashv

Therefore, by Corollary 6.10, the fan functional of model \mathcal{C} is not T -definable from the forall functional, which concludes the proof of Theorem 6.7. In summary:

1. Model \mathcal{C} has fan and forall functionals, but they are not T -definable, and forall is T -definable from fan. To prove the non-definability result, we used submodels of the Scott model consisting of the \mathcal{C} -total and HEO-total functionals.
2. Model \mathcal{M} does not have a fan functional, but does have a forall functional. We defined a submodel of \mathcal{C} , based on the construction of the model \mathcal{M} , to show that the fan functional of \mathcal{C} is not T -definable from the forall functional.
3. Model HEO does not have any of these two functionals. This is well known, but was not used in the above development. What we used, in item (1), is the existence of the Kleene singular predicate in HEO (not definable in T) as an auxiliary tool to establish the failure of T -definability of the forall functional of model \mathcal{C} . This relies on the fact that both \mathcal{C} and HEO can be seen as quotients of submodels of the Scott model.

The non definability results developed in this section refer to the particular model \mathcal{C} of continuous functionals, with the above models and submodels used as tools for the proofs. We applied these results to deduce the non provability of

the existence, or relative existence, of functionals in the language of HA^ω , formulated in a model independent way. This closes some of the gaps in the model independent inter-definability results developed in the previous sections.

Acknowledgements. The authors would like to thank Ulrich Berger for suggesting some improvements on an earlier version of the paper, and Radu Grigore for drawing the diagram of Figure 1 in LaTeX (tikz). The second author also acknowledges support of The Royal Society under grant 516002.K501/RH/kk.

REFERENCES

- [1] J. AVIGAD and S. FEFERMAN, *Gödel's functional ("Dialectica") interpretation*, **Handbook of proof theory** (S. R. Buss, editor), Studies in Logic and the Foundations of Mathematics, vol. 137, North Holland, Amsterdam, 1998, pp. 337–405.
- [2] S. BERARDI, M. BEZEM, and T. COQUAND, *On the computational content of the axiom of choice*, **The Journal of Symbolic Logic**, vol. 63 (1998), no. 2, pp. 600–622.
- [3] U. BERGER, *The Berardi-Bezem-Coquand-functional in a domain-theoretic setting*, Draft, July, 2002.
- [4] U. BERGER and P. OLIVA, *Modified bar recursion and classical dependent choice*, **Lecture Notes in Logic**, vol. 20 (2005), pp. 89–107.
- [5] ———, *Modified bar recursion*, **Mathematical Structures in Computer Science**, vol. 16 (2006), pp. 163–183.
- [6] U. BERGER and H. SCHWICHTENBERG, *Program extraction from classical proofs*, **Logic and computational complexity workshop (LCC'94)** (D. Leivant, editor), Lecture Notes in Computer Science, vol. 960, Springer, Berlin, 1995, pp. 77–97.
- [7] M. BEZEM, *Strongly majorizable functionals of finite type: a model for bar recursion containing discontinuous functionals*, **The Journal of Symbolic Logic**, vol. 50 (1985), pp. 652–660.
- [8] M. H. ESCARDÓ, *Exhaustible sets in higher-type computation*, **Logical Methods in Computer Science**, vol. 4 (2008), no. 3, p. paper 4.
- [9] M. H. ESCARDÓ and P. OLIVA, *Computational interpretations of analysis via products of selection functions*, **Computability in europe 2010, Incs** (F. Ferreira, B. Lowe, E. Mayordomo, and L. M. Gomes, editors), Springer, 2010, pp. 141–150.
- [10] ———, *The Peirce translation and the double negation shift*, **Programs, Proofs, Processes - CiE 2010, LNCS 6158** (F. Ferreira, B. Löwe, E. Mayordomo, and L. M. Gomes, editors), Springer, 2010, pp. 151–161.
- [11] ———, *Selection functions, bar recursion, and backward induction*, **Mathematical Structures in Computer Science**, vol. 20 (2010), no. 2, pp. 127–168.
- [12] F. FERREIRA and P. ENGRÁCIA, *The bounded functional interpretation of the double negation shift*, **The Journal of Symbolic Logic**, vol. 75 (2010), no. 2, pp. 759–773.
- [13] K. GÖDEL, *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes*, **Dialectica**, vol. 12 (1958), pp. 280–287.
- [14] W. A. HOWARD, *Hereditarily majorizable functionals of finite type*, **Metamathematical investigation of intuitionistic Arithmetic and Analysis** (A. S. Troelstra, editor), Lecture Notes in Mathematics, vol. 344, Springer, Berlin, 1973, pp. 454–461.
- [15] U. KOHLENBACH, *Theorie der majorisierbaren und stetigen Funktionale und ihre Anwendung bei der Extraktion von Schranken aus inkonstruktiven Beweisen: Effektive Eindeutigkeitsmodule bei besten Approximationen aus ineffektiven Eindeutigkeitsbeweisen*, **Ph.D. thesis**, Frankfurt, pp. xxii+278, 1990.
- [16] ———, *Applied proof theory: Proof interpretations and their use in mathematics*, Monographs in Mathematics, Springer, 2008.
- [17] D. NORMANN, *The continuous functionals*, **Handbook of computability theory** (E. R. Griffor, editor), North Holland, Amsterdam, 1999, pp. 251–275.
- [18] C. SPECTOR, *Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics*, **Recursive function theory: Proc. symposia in pure mathematics** (F. D. E. Dekker, editor), vol. 5, American Mathematical Society, Providence, Rhode Island, 1962, pp. 1–27.

- [19] A. S. TROELSTRA, *Metamathematical investigation of intuitionistic arithmetic and analysis*, Lecture Notes in Mathematics, vol. 344, Springer, Berlin, 1973.