Bar recursive encodings of tree ordinals

Marc Bezem
Utrecht University *

Wilfried Buchholz
Ludwig-Maximilians-Universität †

1. Introduction

We ask the attention for a definition schema from higher order subrecursion theory called *bar recursion*. Bar recursion originates with Spector [12], where bar recursion of all finite types is shown to characterize the class of provably total recursive functions of analysis. This class has also been characterized by Girard [3] as those functions which are definable in the second order typed lambda calculus ($\lambda 2$, or the *polymorphic* lambda calculus). Analysis is a so-called *impredicative* theory since it allows defining a predicate by quantification over all predicates, including the predicate that is defined itself. In $\lambda 2$ terms may depend on types, including the type of the term itself. This phenomenon is also called impredicativity. Impredicativity is an informal notion which has many faces. The question arises: is there something impredicative about bar recursion? The question is important for a better mathematical understanding of impredicativity, especially of its computational impact. We answer the question affirmatively by identifying and exploring a special feature of bar recursion which we consider to be another appearance of impredicativity: the dependency of terms on *order* types.

Observe that in the case of (higher order) primitive recursion the order type ω of the recursion is fixed. In the case of bar recursion, however, there is a parameter which determines the order type of the recursion. Lambda abstraction from this parameter is perfectly allowed, thus yielding a polymorphic—or better: a *poly-order-type*—recursor. We explore the computational aspects of the dependency of terms on order types to some extent, by constructing bar recursive encodings of transfinite recursors up to ε_0 and beyond. Transfinite recursors are of some interest for computer science: they provide a strongly normalizing formalism for recursive functional programming. Thus program termination is guaranteed for all evaluation strategies; this is not the case when using fixed point combinators.

*Department of Philosophy, P.O. Box 80126, 3508 TC Utrecht, The Netherlands, e-mail: bezem@phil.ruu.nl.

†Mathematisches Institut, Theresienstr. 39, D-8000 München 2, Germany, e-mail: buchholz@mathematik.uni-muenchen.dbp.de.

At present it is unknown what the exact limitations of the methods of this paper are. It should be remarked that we are only using bar recursion of lowest type. It is not obvious how the generalization to bar recursion of higher type should be done. For metamathematical reasons the overall limitation of such generalizations is given by the (largely unknown) so-called ordinal of analysis, i.e. the ordinal which is related to analysis in the same way as ε_0 is related to arithmetic.

Most of the material presented in this paper can be found elsewhere in the literature, though sometimes in a different presentation. A paper which is particularly close to ours is Vogel [14], which came to our notice only in the final stage of the completion of this paper. We have tried to give proper credits and adequate references.

2. Preliminaries

2.1. Type structure

For sets $\mathcal{A}, \mathcal{B}$ we denote the set of all mappings from $\mathcal{A}$ to $\mathcal{B}$ by $\mathcal{A}\to\mathcal{B}$. If $f : \mathcal{A}\to\mathcal{A}$, then the n-th iteration of f, denoted by $f^n : \mathcal{A}\to\mathcal{A}$ ($n \in \mathcal{N}$), is defined recursively by $f^0(a) = a$, $f^{n+1}(a) = f(f^n(a))$ for all $a \in \mathcal{A}$.

Types are those of the simply typed lambda calculus, i.e. 0 and with τ and τ' also $\tau\to\tau'$. We write 1 for the type $0\to0$, 2 for $1\to0$, and so on. The *level* of a type is inductively defined as follows: $\mathrm{level}(0) = 0$; $\mathrm{level}(\tau\to\tau') = \max(1 + \mathrm{level}(\tau), \mathrm{level}(\tau'))$.

In order to avoid cumbersome syntactic details we prefer to explain our ideas in a set-theoretical, semantical framework. Throughout this paper we work in a convenient extensional model for bar recursion, e.g. the model $\mathcal{M} = \bigcup \mathcal{M}_\tau$ from Bezem [1]. For the purpose of this paper it is not necessary to reproduce the precise definition of $\mathcal{M}$. It suffices to stipulate the following: (i) $\mathcal{M}_0$ is $\mathcal{N}$, the set of natural numbers; (ii) $\mathcal{M}_{0\to\tau} = \mathcal{M}_0\to\mathcal{M}_\tau$ for all τ; (iii) $\mathcal{M}_{\tau\to\tau'}$ is a specific proper subset of $\mathcal{M}_\tau\to\mathcal{M}_{\tau'}$ for all $\tau \neq 0$; (iv) $\mathcal{M}$ is a model of the simply typed lambda calculus containing primitive recursors and bar recursors of all appropriate types. It is useful to remark that if we equip $\mathcal{M}_1 = \mathcal{N}\to\mathcal{N}$ with the Baire topology and $\mathcal{N}$ with the discrete topology, then all continuous mappings of $\mathcal{N}\to\mathcal{N}$ to $\mathcal{N}$ are in $\mathcal{M}_2$. (The converse does not hold: $\mathcal{M}_2$ contains discontinuous functionals.)

Elements of $\mathcal{M}$ ($\mathcal{M}_\tau$) are called *functionals* (of type τ) and will be denoted by $F, G, H, \ldots$. However, functionals of type 0, i.e. natural numbers, will be denoted by $n, m, x, y, \ldots$. The result of applying a functional F of type $\tau\to\tau'$ to a functional G of type τ is a functional of type τ' denoted by FG. In expressions like FGH association is assumed to be to the left and the functionals are assumed to have appropriate types.

2.2. Primitive recursion and bar recursion

A *primitive recursor* is a functional $\boldsymbol{R}_\tau$ satisfying

$$\boldsymbol{R}_\tau MN0 = M \text{ and } \boldsymbol{R}_\tau MN(n+1) = Nn(\boldsymbol{R}_\tau MNn)$$

for all M, N, n of appropriate types. The type of $\boldsymbol{R}_\tau$ is determined by the type τ of M, but we shall ignore this type and simply write $\boldsymbol{R}$, relying on the assumption that all expressions are well-typed. A *primitive recursive functional* is a functional that can be constructed from $0 \in \mathcal{M}_0$, the successor function $\boldsymbol{S} = \lambda n.n+1 \in \mathcal{M}_1$ and primitive recursors of all appropriate types, using lambda abstraction and application. Thus the term 'primitive recursion' includes throughout this paper what is usually termed 'higher order primitive recursion'. We freely use primitive recursion without specifying the primitive recursive functionals in question.

Let $\mathcal{N}^*$ be the set of finite sequences of natural numbers. We let $\sigma, \sigma', \ldots$ range over $\mathcal{N}^*$. If $\sigma = \langle n_0, \ldots, n_{k-1}\rangle$ $(k \geq 0)$, then $\mathrm{lh}(\sigma) = k$ denotes the length of σ and, for $i < \mathrm{lh}(\sigma)$, $\sigma_i = n_i$ denotes the i-th projection of σ. Concatenation of finite sequences is expressed by $*$. As usual $\sigma \preceq \sigma'$ is defined by $\exists \sigma''\ \sigma * \sigma'' = \sigma'$ and $\sigma \prec \sigma'$ by $\sigma \preceq \sigma' \ \wedge \ \sigma \neq \sigma'$. We define $[\sigma]$ to be the function assigning σ_i to $i < \mathrm{lh}(\sigma)$ and 0 to $i \geq \mathrm{lh}(\sigma)$. We abbreviate $\langle x\rangle * \sigma$ by $x * \sigma$ and $\sigma * \langle x\rangle$ by $\sigma * x$. Furthermore we abbreviate $\{x * \sigma \mid \sigma \in \Sigma\}$ by $x * \Sigma$ and $\{\sigma * x \mid \sigma \in \Sigma\}$ by $\Sigma * x$ for all sets of finite sequences Σ. Finally, $\Sigma * \Sigma'$ abbreviates $\{\sigma * \sigma' \mid \sigma \in \Sigma,\ \sigma' \in \Sigma'\}$.

We assume a surjective, primitive recursive encoding of finite sequences by natural numbers, with $\mathrm{lh}(\sigma)$, σ_i, $*$, $\prec$, $[\sigma]$ primitive recursive. It will always be clear when a natural number should be taken as the code of a finite sequence. The partial ordering $\prec$ structures $\mathcal{N}^*$ into a (infinitely branching, non well-founded) tree. The set of paths of this tree is $\mathcal{N} \to \mathcal{N}$, which coincides with $\mathcal{M}_1$. We shall be concerned with well-founded subsets of $\mathcal{N}^*$.

A *bar recursor* is a functional $\boldsymbol{B}_{0,\tau}$ satisfying

$$\boldsymbol{B}_{0,\tau} YGH\sigma = \begin{cases} G\sigma & \text{if } Y[\sigma] \leq \mathrm{lh}(\sigma) \\ H(\lambda x.\boldsymbol{B}_{0,\tau} YGH(\sigma * x))\sigma & \text{otherwise} \end{cases}$$

for all Y, G, H, σ of appropriate types. In this paper we restrict ourselves to bar recursion of type 0 (whence $\boldsymbol{B}_{0,\tau}$ instead of $\boldsymbol{B}_{\tau',\tau}$), i.e. the finite sequences consist of natural numbers (functionals of type 0, instead of functionals of type τ'). Thus the type of a bar recursor $\boldsymbol{B}_{0,\tau}$ is determined by the type τ of $G\sigma$. More specifically, Y has type 2, G has type $0 \to \tau$ (since finite sequences are encoded as natural numbers) and H has type $(0 \to \tau) \to 0 \to \tau$. Again we shall ignore all these types and simply write $\boldsymbol{B}$, relying on the assumption that all expressions are well-typed. A *bar recursive functional* is a functional that can be constructed from 0, $\boldsymbol{S}$, primitive recursors $\boldsymbol{R}_\tau$ and bar recursors $\boldsymbol{B}_{0,\tau}$ for all τ, using lambda abstraction and application. Let BR (BR_τ) denote the set of bar recursive functionals (of type τ).

Bar recursion is in fact recursion on trees of finite sequences. For example, for fixed Y, G, H the value of $\boldsymbol{B}YGH\langle\rangle$ is determined by recursion on the tree $\{\sigma \mid \forall \sigma' \prec \sigma\ Y[\sigma'] > \mathrm{lh}(\sigma')\}$. Of course Y should be such that this tree is well-founded. Observe that this is not true for all mappings $Y : \mathcal{M}_1 \to \mathcal{M}_0$, for example take Y defined by $YF = n+1$ if $(\forall i < n\ (Fi = 1)) \wedge (\forall i \geq n\ (Fi = 0))$, and $YF = 0$ if no such n exists. This shows that it is non-trivial to construct a model for bar recursion (recall that $\mathcal{M}_{\tau\to\tau'}$ is a *proper* subset of $\mathcal{M}_\tau \to \mathcal{M}_{\tau'}$ for $\tau \neq 0$). The reasons for the specific choice of the termination condition $Y[\sigma] \leq \mathrm{lh}(\sigma)$ in the schema of bar recursion are subtle. At first sight the way in which Y determines the subtree of $\mathcal{N}^*$ for the computation of $\boldsymbol{B}YGH\sigma$, vz. $\{\sigma * \sigma' \mid \forall \sigma'' \prec \sigma'\ Y[\sigma * \sigma''] > \mathrm{lh}(\sigma * \sigma'')\}$, seems rather indirect. However, there are two reasons for this. First, the above subtrees are primitive recursive in Y. Second, despite the counterexample above, the set of functionals Y for which all such trees are well-founded is very large and contains subsets (such as $\mathcal{M}_2$) satisfying important closure properties, in particular closure under bar recursion.

2.3. Tree ordinals

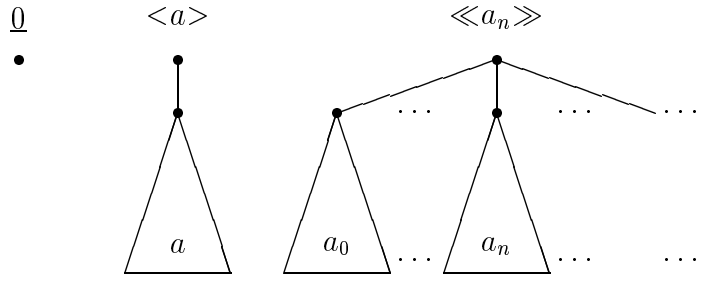
In this paper we shall frequently refer to standard ordinal theory. The small part of ordinal theory we use (ordinal arithmetic, Cantor normal form, normal function and the Veblen hierarchy) is completely covered by, for example, Pohlers [8]. Let ON_0 be the set of all countable ordinals. Ordinals will be denoted by $0, 1, \ldots, \omega, \ldots, \varepsilon_0$ etc. (ε_0 is the least fixpoint of $\lambda\alpha.\,\omega^\alpha$), with variables $\alpha, \beta, \alpha', \ldots$ ranging over ON_0. We use a standard representation of countable ordinals by well-founded (countable) trees, also called (countable) tree ordinals and denoted by TO_0.

2.1. Definition. The set TO_0 is inductively defined by:

(i) $\underline{0} \in \mathrm{TO}_0$;
(ii) $a \in \mathrm{TO}_0 \Rightarrow <a> \in \mathrm{TO}_0$;
(iii) $(\forall n \in \mathcal{N}\ a_n \in \mathrm{TO}_0) \Rightarrow \ll a_n \gg_{n\in\mathcal{N}} \in \mathrm{TO}_0$.

Here and below $a, a', a_n, b, \ldots$ range over TO_0. Furthermore, $\underline{n}$ abbreviates $<..<\underline{0}>..>$ (nesting of depth n) for all n, and $\underline{\omega}$ denotes $\ll\underline{n}\gg$. We allow ourselves the somewhat loose notation $\ll a_n \gg$ instead of $\ll a_n \gg_{n\in\mathcal{N}}$. Ambiguity such as in $\ll a_{n+m} \gg$ will never occur.

Tree ordinals are best imagined by their graphical representation:



The usual ordinal arithmetic can be defined in terms of tree ordinals; we rephrase these definitions in order to be self-contained.

2.2. DEFINITION. Addition, multiplication and exponentiation on TO_0 are inductively defined as follows:

(i) $a + \underline{0} = a$; $a + <b> = <a + b>$; $a + \ll b_n \gg = \ll a + b_n \gg$;
(ii) $a \cdot \underline{0} = \underline{0}$; $a \cdot <b> = (a \cdot b) + a$; $a \cdot \ll b_n \gg = \ll a \cdot b_n \gg$;
(iii) $a^{\underline{0}} = \underline{1}$; $a^{<b>} = a^b \cdot a$; $a^{\ll b_n \gg} = \ll a^{b_n} \gg$.

Note that $a + b$ is in fact b with all occurrences of $\underline{0}$ replaced by a. In graphical terms: the tree of $a + b$ consists of the tree of b with all the leaves replaced by the tree of a.

The relation between tree ordinals and ordinals is made precise by the following notion of height of a tree ordinal.

2.3. DEFINITION. The *height* of $a \in \mathrm{TO}_0$, denoted by $|a|$, is the ordinal inductively defined by: $|\underline{0}| = 0$; $|<a>| = |a| + 1$; $|\ll a_n \gg| = \sup\{|a_n| + 1 \mid n \in \mathcal{N}\}$.

2.4. DIGRESSION. It is tempting to define: $a \lhd b$ iff $|a| < |b|$. Indeed $\lhd$ is a strict partial order on TO_0. There are two natural ways to extend $\lhd$ to a reflexive relation: $a \trianglelefteq b$ iff $a \lhd b \vee |a| = |b|$, and $a \mathrel{\overline{\lhd}} b$ iff $a \lhd b \vee a = b$. The latter is indeed a partial ordering, but not complete: $\mathcal{X} = \{\underline{n} \mid n \in \mathcal{N}\}$ has no supremum in the sense of $\overline{\lhd}$, since $\underline{\omega}$ and $\underline{1} + \underline{\omega}$ are distinct minimal upper bounds of $\mathcal{X}$. The former, however, is not an ordering, but a preorder (reflexive and transitive): $\underline{\omega} \trianglelefteq \underline{1} + \underline{\omega} \trianglelefteq \underline{\omega}$, but $\underline{\omega} \neq \underline{1} + \underline{\omega}$ (the trees are different, although they have the same height). Dividing out $|=|$ defined by $a \mid=\mid b$ iff $|a| = |b|$ would result in a structure $(\mathrm{TO}_0/\ |=|, \trianglelefteq)$ which is order isomorphic with $(\mathrm{ON}_0, \leq)$. For our purpose, however, it is better to keep working with the trees and not with the equivalence classes.

3. Encoding tree ordinals as type 2 functionals

Encoding tree ordinals a as type 2 functionals Y_a is unconventional. Usually, encodings of ordinals are primitive recursive relations on $\mathcal{N}$ and the proof that this relation is a well-ordering requires *proof theoretic* strength. Our encodings so to say carry their own proof, and as a consequence the encoding requires *computational* strength. Given Y_a, transfinite recursion with respect to a can be encoded as $\boldsymbol{B}Y_aGH$ for suitable G, H. In this section we work out the encoding in all detail.

The following definition of functionals Y_a should be understood with the graphical representation of the tree ordinals in mind. The functions $F \in \mathcal{M}_1 = \mathcal{N} \to \mathcal{N}$ stand for paths in the tree $\mathcal{N}^*$. The function $F \leftarrow 1$ is 'F shifted one position to the left', so considering the path F relative to the subtree one level

lower, with root $\langle F0\rangle$. The lower level of $\langle F0\rangle$ with respect to $\langle\rangle$ is compensated for by '1+'.

3.1. Definition. For $F \in \mathcal{M}_1$, and $x \in \mathcal{N}$ we define:

$$F{\leftarrow}x = \lambda y.\, F(x+y)$$

By induction on $a \in \mathrm{TO}_0$ we define type 2 functionals Y_a:

$$\begin{array}{rcl} Y_{\underline{0}} & = & \lambda F.\, 0 \\ Y_{<a>} & = & \lambda F.\, \mathsf{if}\ F0 = 0\ \mathsf{then}\ 1 + Y_a(F{\leftarrow}1)\ \mathsf{else}\ 0 \\ Y_{\ll a_n \gg} & = & \lambda F.\, 1 + Y_{a_{F0}}(F{\leftarrow}1) \end{array}$$

It can easily be proved by induction on a that Y_a is a continuous mapping from $\mathcal{N}{\to}\mathcal{N}$ to $\mathcal{N}$ for every $a \in \mathrm{TO}_0$. As a consequence $Y_a \in \mathcal{M}_2$ for every $a \in \mathrm{TO}_0$.

The primitive recursive predicates $\mathrm{Bartree}(Y,\sigma)$ and $\mathrm{Ortree}(Y,\sigma)$ defined below isolate subtrees of $\mathcal{N}^*$. The former isolates the tree for the computation of $\boldsymbol{B}YGH\langle\rangle$ discussed in Subsection 2.2. The latter isolates the subset of the former that represents the tree ordinal a in case $Y = Y_a$ ('Ortree' stands for ordinal tree). Thus Y_a encodes via the ordinal tree $\mathrm{Ortree}(Y_a)$ the tree ordinal $a \in \mathrm{TO}_0$. The primitive recursive predicates $\mathrm{Zero}(Y,\sigma)$, $\mathrm{Succ}(Y,\sigma)$ and $\mathrm{Join}(Y,\sigma)$ represent the corresponding properties of ordinal (sub)trees with root σ, denoted by $\mathrm{Ortree}(Y){\downarrow}\sigma$.

3.2. Definition. For $Y \in \mathcal{M}_2$ and $\sigma \in \mathcal{N}^*$ we define:

$$\begin{array}{rcl} \mathrm{Bartree}(Y,\sigma) & \Leftrightarrow & \forall \sigma' \prec \sigma\ Y[\sigma'] > \mathrm{lh}(\sigma') \\ \mathrm{Bartree}(Y) & = & \{\sigma \mid \mathrm{Bartree}(Y,\sigma)\} \\ \mathrm{Ortree}(Y,\sigma) & \Leftrightarrow & \mathrm{Bartree}(Y,\sigma) \wedge Y[\sigma] \geq \mathrm{lh}(\sigma) \\ \mathrm{Ortree}(Y) & = & \{\sigma \mid \mathrm{Ortree}(Y,\sigma)\} \\ \mathrm{Ortree}(Y){\downarrow}\sigma & = & \{\sigma' \mid \sigma \star \sigma' \in \mathrm{Ortree}(Y)\} \\ \mathrm{Zero}(Y,\sigma) & \Leftrightarrow & \mathrm{Ortree}(Y,\sigma) \wedge \neg\mathrm{Ortree}(Y,\sigma{*}0) \\ \mathrm{Succ}(Y,\sigma) & \Leftrightarrow & \mathrm{Ortree}(Y,\sigma) \wedge \mathrm{Ortree}(Y,\sigma{*}0) \wedge \neg\mathrm{Ortree}(Y,\sigma{*}1) \\ \mathrm{Join}(Y,\sigma) & \Leftrightarrow & \mathrm{Ortree}(Y,\sigma) \wedge \mathrm{Ortree}(Y,\sigma{*}0) \wedge \mathrm{Ortree}(Y,\sigma{*}1) \end{array}$$

The following lemmas are instrumental to prove that the tree ordinals a from Section 2.3 are isomorphic to $\mathrm{Ortree}(Y_a)$ and that $\mathrm{Zero}(Y_a,\sigma)$, $\mathrm{Succ}(Y_a,\sigma)$ and $\mathrm{Join}(Y_a,\sigma)$ are correctly representing the corresponding notions for tree ordinals represented by $\mathrm{Ortree}(Y_a){\downarrow}\sigma$.

3.3. Lemma.

(i) Bartree, Ortree, Zero, Succ *and* Join *are primitive recursive*

(ii) $\mathrm{Zero}(Y,\sigma) \Rightarrow (Y[\sigma] = \mathrm{lh}(\sigma) \wedge \neg\mathrm{Succ}(Y,\sigma) \wedge \neg\mathrm{Join}(Y,\sigma))$
(iii) $\mathrm{Succ}(Y,\sigma) \Rightarrow (Y[\sigma] > \mathrm{lh}(\sigma) \wedge \neg\mathrm{Zero}(Y,\sigma) \wedge \neg\mathrm{Join}(Y,\sigma))$
(iv) $\mathrm{Join}(Y,\sigma) \Rightarrow (Y[\sigma] > \mathrm{lh}(\sigma) \wedge \neg\mathrm{Zero}(Y,\sigma) \wedge \neg\mathrm{Succ}(Y,\sigma))$
(v) $\mathrm{Ortree}(Y,\sigma) \Leftrightarrow (\mathrm{Zero}(Y,\sigma) \vee \mathrm{Succ}(Y,\sigma) \vee \mathrm{Join}(Y,\sigma))$

PROOF. (i) Obvious. (ii) Assume $\mathrm{Zero}(Y,\sigma)$, so $\mathrm{Ortree}(Y,\sigma) \wedge \neg\mathrm{Ortree}(Y,\sigma{*}0)$, so $\mathrm{Bartree}(Y,\sigma) \wedge Y[\sigma] \geq \mathrm{lh}(\sigma) \wedge (\neg\mathrm{Bartree}(Y,\sigma{*}0) \vee Y[\sigma{*}0] < \mathrm{lh}(\sigma{*}0))$. By the definition of Bartree and by using $[\sigma{*}0] = [\sigma]$ it follows in both cases of the disjunction that $Y[\sigma] = \mathrm{lh}(\sigma)$. The mutual exclusion of the predicates Zero, Succ and Join is immediate. (iii),(iv) Assume $\mathrm{Succ}(Y,\sigma)$ or $\mathrm{Join}(Y,\sigma)$. Then $\mathrm{Ortree}(Y,\sigma{*}0)$, so $\mathrm{Bartree}(Y,\sigma{*}0)$ and hence $Y[\sigma] > \mathrm{lh}(\sigma)$. (v) Immediate from the definitions. □

3.4. LEMMA.
(i) $\mathrm{Bartree}(Y_{\underline{0}}) = \mathrm{Ortree}(Y_{\underline{0}}) = \{\langle\rangle\}$
(ii) $\mathrm{Zero}(Y_{\underline{0}}, \langle\rangle)$

PROOF. Obvious. □

3.5. LEMMA. *For all $a \in \mathrm{TO}_0$ we have:*
(i) $\mathrm{Bartree}(Y_{<a>}) = \{\langle\rangle\} \cup \{\langle x\rangle \mid x > 0\} \cup 0{*}\mathrm{Bartree}(Y_a)$
(ii) $\mathrm{Ortree}(Y_{<a>}) = \{\langle\rangle\} \cup 0{*}\mathrm{Ortree}(Y_a)$
(iii) $\mathrm{Succ}(Y_{<a>}, \langle\rangle)$
(iv) $\boxed{\cdots}(Y_{<a>}, 0{*}\sigma) \Leftrightarrow \boxed{\cdots}(Y_a, \sigma)$ *for* $\boxed{\cdots} \in \{\mathrm{Zero},\mathrm{Succ},\mathrm{Join}\}$

PROOF. (i) Follows immediately from the definition of Bartree and $Y_{<a>}$. (ii,⇒) Assume $\sigma \in \mathrm{Ortree}(Y_{<a>})$ with $\sigma \neq \langle\rangle$. Then we have $0 < \mathrm{lh}(\sigma) \leq Y_{<a>}[\sigma] = \mathsf{if}\ \sigma_0 = 0\ \mathsf{then}\ 1 + Y_a[\langle\sigma_1,\ldots,\sigma_{\mathrm{lh}(\sigma)-1}\rangle]\ \mathsf{else}\ 0$. It follows that $\sigma_0 = 0$ and

$$Y_a[\langle\sigma_1,\ldots,\sigma_{\mathrm{lh}(\sigma)-1}\rangle] \geq \mathrm{lh}(\sigma) - 1 = \mathrm{lh}(\langle\sigma_1,\ldots,\sigma_{\mathrm{lh}(\sigma)-1}\rangle).$$

By (i) we have $\langle\sigma_1,\ldots,\sigma_{\mathrm{lh}(\sigma)-1}\rangle \in \mathrm{Bartree}(Y_a)$, so $\langle\sigma_1,\ldots,\sigma_{\mathrm{lh}(\sigma)-1}\rangle \in \mathrm{Ortree}(Y_a)$. (ii,⇐) Obviously $\langle\rangle \in \mathrm{Ortree}(Y_{<a>})$. Furthermore, if $\sigma \in \mathrm{Ortree}(Y_a)$, then by definition $\sigma \in \mathrm{Bartree}(Y_a)$ and $Y_a[\sigma] \geq \mathrm{lh}(\sigma)$. It follows by (i) that $0{*}\sigma \in \mathrm{Bartree}(Y_{<a>})$ and $Y_{<a>}[0{*}\sigma] = 1 + Y_a[\sigma] \geq 1 + \mathrm{lh}(\sigma) = \mathrm{lh}(0{*}\sigma)$. Hence $0{*}\sigma \in \mathrm{Ortree}(Y_{<a>})$. (iii),(iv) Follow from (ii) by the way in which Zero, Succ and Join are defined in terms of Ortree. □

3.6. LEMMA. *For all $\ll a_n \gg \in \mathrm{TO}_0$ we have:*
(i) $\mathrm{Bartree}(Y_{\ll a_n \gg}) = \{\langle\rangle\} \cup \bigcup_{x\in\mathcal{N}} x{*}\mathrm{Bartree}(Y_{a_x})$
(ii) $\mathrm{Ortree}(Y_{\ll a_n \gg}) = \{\langle\rangle\} \cup \bigcup_{x\in\mathcal{N}} x{*}\mathrm{Ortree}(Y_{a_x})$
(iii) $\mathrm{Join}(Y_{\ll a_n \gg}, \langle\rangle)$
(iv) $\boxed{\cdots}(Y_{\ll a_n \gg}, x{*}\sigma) \Leftrightarrow \boxed{\cdots}(Y_{a_x}, \sigma)$ *for* $\boxed{\cdots} \in \{\mathrm{Zero},\mathrm{Succ},\mathrm{Join}\}$

PROOF. Analogous to the proof of the previous lemma. □

3.7. ISOMORPHY LEMMA. *Every* $a \in \mathrm{TO}_0$ *is isomorphic to* $\mathrm{Ortree}(Y_a)$.

PROOF. By induction on $a \in \mathrm{TO}_0$, using the lemmas 3.4(i), 3.5(ii) and 3.6(ii). □

From now on we shall identify the tree ordinal a with the ordinal tree $\mathrm{Ortree}(Y_a)$. For all $a \in \mathrm{TO}_0$ the ordinal subtree $\mathrm{Ortree}(Y_a)\!\downarrow\!\sigma$ also represents an element of TO_0, in fact a subtree of a, denoted by $a\!\downarrow\!\sigma$. In other words, the finite sequence $\sigma \in \mathrm{Ortree}(Y_a)$ is decoded into $a\!\downarrow\!\sigma \in \mathrm{TO}_0$. We end this section by the following lemma stating that the predicates $\mathrm{Zero}(Y,\sigma)$, $\mathrm{Succ}(Y,\sigma)$ and $\mathrm{Join}(Y,\sigma)$ are adequate.

3.8. LEMMA. *For all* $a \in \mathrm{TO}_0$ *and* $\sigma \in \mathrm{Ortree}(Y_a)$ *we have:*

(i) $\exists x > 0\ (\sigma{*}x \in \mathrm{Ortree}(Y_a)) \Rightarrow \forall x > 0\ (\sigma{*}x \in \mathrm{Ortree}(Y_a))$
(ii) $\mathrm{Zero}(Y_a,\sigma) \Leftrightarrow a\!\downarrow\!\sigma = \underline{0}$
(iii) $\mathrm{Succ}(Y_a,\sigma) \Leftrightarrow a\!\downarrow\!\sigma = <a\!\downarrow\!\sigma{*}0>$
(iv) $\mathrm{Join}(Y_a,\sigma) \Leftrightarrow a\!\downarrow\!\sigma = \ll a\!\downarrow\!\sigma{*}n \gg$

PROOF. (i) By induction on $a \in \mathrm{TO}_0$, using the lemmas 3.4, 3.5 and 3.6. (ii)-(iv) By previous lemmas it follows that $\mathrm{Zero}(Y_a,\sigma) \Leftrightarrow \mathrm{Ortree}(Y_a)\!\downarrow\!\sigma = \{\sigma\}$, $\mathrm{Succ}(Y_a,\sigma) \Leftrightarrow \mathrm{Ortree}(Y_a)\!\downarrow\!\sigma = \{\sigma\} \cup \mathrm{Ortree}(Y_a)\!\downarrow\!\sigma{*}0$, and

$$\mathrm{Join}(Y_a,\sigma) \Leftrightarrow \mathrm{Ortree}(Y_a)\!\downarrow\!\sigma = \{\sigma\} \cup \bigcup_{x\in\mathcal{N}} \mathrm{Ortree}(Y_a)\!\downarrow\!\sigma{*}x.$$

Now (ii)-(iv) follow by using (i). □

4. Bar recursive construction of some Y_a

4.1. Transfinite iterations

4.1. DEFINITION. Let Z, S, J be functionals of types $\tau, \tau{\to}\tau, (0{\to}\tau){\to}\tau$, respectively. By induction on $a \in \mathrm{TO}_0$ we define the a-iteration of J (join case) and S (successor case) on Z (zero case), denoted by $(J;S)^a Z$:

$$\begin{aligned} (J;S)^{\underline{0}} Z &= Z \\ (J;S)^{<a>} Z &= S((J;S)^a Z) \\ (J;S)^{\ll a_n \gg} Z &= J(\lambda n.\,(J;S)^{a_n} Z) \end{aligned}$$

Note that $(J;S)^{\underline{n}} Z = S^n Z$ for all $n \in \mathcal{N}$.

Next we show that $(J;S)^a Z$ is bar recursive in Y_a.

4.2. Definition. We define the following bar recursive functional $\boldsymbol{I}$ (for iterator, not for identity; $\boldsymbol{I}$ is in fact a poly-order-type iterator):

$$\begin{array}{rcl} \boldsymbol{I} & = & \lambda YZSJ.\, \boldsymbol{B}YG_ZH_{Y,S,J}, \text{ where} \\ G_Z & = & \lambda\sigma.\, Z \text{ and} \\ H_{Y,S,J} & = & \lambda A\sigma.\ \mathsf{if}\ \ \mathrm{Succ}(Y,\sigma)\ \ \mathsf{then}\ \ S(A0) \\ & & \quad\ \mathsf{elsif}\ \ \mathrm{Join}(Y,\sigma)\ \ \mathsf{then}\ \ JA \\ & & \quad\ \mathsf{else}\ \ 0 \end{array}$$

Note that G_Z and $H_{Y,S,J}$ are primitive recursive.

4.3. Theorem. *For all $a \in \mathrm{TO}_0$ and $\sigma \in \mathrm{Ortree}(Y_a)$ we have:*

$$(J;S)^{a\downarrow\sigma}Z = \boldsymbol{I}Y_aZSJ\sigma$$

Proof. By induction on $a\downarrow\sigma \in \mathrm{TO}_0$, using Lemma 3.8. □

4.2. Construction of $Y_{\underline{\omega}^a}$ from Y_a

The construction of $Y_{\underline{\omega}^a}$ from Y_a is done in two steps. First, we construct a primitive recursive functional P such that $PY_{\underline{\omega}^a} = Y_{\underline{\omega}^{<a>}}$ for all $a \in \mathrm{TO}_0$. Second, we construct a bar recursive functional Q which transfinitely iterates P. In Q the phenomenon of terms depending on order types can be observed, as it is using the poly-order-type iterator $\boldsymbol{I}$ from Definition 4.2.

4.4. Lemma. *There exists a primitive recursive functional P such that $PY_a = Y_{a\cdot\underline{\omega}}$ for all $a \in \mathrm{TO}_0$.*

Proof. Let $a \in \mathrm{TO}_0$. By $a \cdot \underline{\omega} = \ll a \cdot \underline{n} \gg$ we have

$$Y_{a\cdot\underline{\omega}} = \lambda F.\, 1 + Y_{a\cdot F0}(F{\leftarrow}1).$$

Hence it suffices to show that $Y_{a\cdot\underline{n}}$ is primitive recursive in Y_a. This can be seen as follows. By the definition of tree ordinal addition we have

$$Y_{a+b} = \lambda F.\, Y_bF + Y_a(F{\leftarrow}Y_bF).$$

So we have the following recursion equation:

$$Y_{a\cdot\underline{n+1}} = \lambda F.\, Y_aF + Y_{a\cdot\underline{n}}(F{\leftarrow}Y_aF)$$

yielding that $Y_{a\cdot\underline{n}}$ is primitive recursive in Y_a. □

4.5. Lemma. *There exists a bar recursive functional Q such that $QY_a = Y_{\underline{\omega}^a}$ for all $a \in \mathrm{TO}_0$.*

PROOF. Define

$$\begin{aligned} Z &= Y_{\underline{\omega}^0}, \\ S &= P \qquad \text{from Lemma 4.4, and} \\ J &= \lambda XF.\, 1 + X(F0)(F{\leftarrow}1). \end{aligned}$$

Then it follows by induction on $a \in \mathrm{TO}_0$ that $Y_{\underline{\omega}^a} = (J; S)^a Z$. So by Theorem 4.3 we have for $Q = \lambda Y.\, \boldsymbol{I} Y Z S J \langle\rangle$ that $QY_a = Y_{\underline{\omega}^a}$. $\square$

4.3. Construction of $Y_{\underline{\varepsilon}_0}$

The standard way to approximate ε_0 is to iterate the ordinal function $\lambda\alpha.\, \omega^\alpha$ whose least fixpoint defines ε_0. We define the tree ordinal $\underline{\varepsilon}_0$ with $|\underline{\varepsilon}_0| = \varepsilon_0$ in a similar way:

$$\underline{\varepsilon}_0 = {\ll}(\lambda a.\, \underline{\omega}^a)^n(\underline{0}){\gg}$$

The functional Q from Lemma 4.5 can be regarded as the representation of $\lambda a.\underline{\omega}^a$ on the encodings of tree ordinals.

4.6. DEFINITION. A functional A of type $2{\to}2$ *represents* a function $f : \mathrm{TO}_0{\to}\mathrm{TO}_0$ if $AY_a = Y_{f(a)}$ for all $a \in \mathrm{TO}_0$.

As examples we mention that P from Lemma 4.4 represents $\lambda a.\, a \cdot \underline{\omega}$ and Q from Lemma 4.5 represents $\lambda a.\, \underline{\omega}^a$. Moreover, if we define $S_{<\cdot>} = \lambda YF.$ if $F0 = 0$ then $1 + Y(F{\leftarrow}1)$ else 0 then $S_{<\cdot>}$ represents the successor function $\lambda a.\, {<}a{>}$.

We now express the type 2 functional $Y_{\underline{\varepsilon}_0}$ encoding $\underline{\varepsilon}_0$ primitive recursively in a functional representing $\lambda a.\, \underline{\omega}^a$.

4.7. THEOREM. *Let $Y_{\underline{\varepsilon}_0}$ be defined according to Definition 3.1 and let Q represent $\lambda a.\, \underline{\omega}^a$. Then we have:*

$$Y_{\underline{\varepsilon}_0} = \lambda F.\, 1 + Q^{F0} Y_{\underline{0}}(F{\leftarrow}1)$$

PROOF. Define $\underline{\varepsilon}_0[n] = (\lambda a.\underline{\omega}^a)^n(\underline{0})$, then $\underline{\varepsilon}_0 = {\ll}\underline{\varepsilon}_0[n]{\gg}$. Let Q represent $\lambda a.\underline{\omega}^a$. By induction one proves for all $n \in \mathcal{N}$:

$$Q^n Y_{\underline{0}} = Y_{\underline{\varepsilon}_0[n]}.$$

It follows that $\lambda F.\, 1 + Q^{F0} Y_{\underline{0}}(F{\leftarrow}1) = \lambda F.\, 1 + Y_{\underline{\varepsilon}_0[F0]}(F{\leftarrow}1) = Y_{\underline{\varepsilon}_0}$. $\square$

4.8. COROLLARY. $Y_{\underline{\varepsilon}_0} = \lambda F.\, 1 + Q^{F0} Y_{\underline{0}}(F{\leftarrow}1)$ *with Q from Lemma 4.5.*

5. Beyond ε_0

It is neither difficult nor exciting to go beyond $\underline{\varepsilon}_0$ constructing $Y_{<\underline{\varepsilon}_0>}, \ldots$. More interesting is the following generalization of the methods of the previous sections of this paper. The idea turned out to be similar as in Vogel [14], who has the priority. Upper ordinal bounds are obtained by Howard [6] for cases in which the bar recursor $\boldsymbol{B}_{0,\tau}$ has a type level 3 or 4, which means that τ has level at most 2. (Recall that the *level* of a type is defined by: $\mathrm{level}(0) = 0$; $\mathrm{level}(\tau{\to}\tau') = \max(1 + \mathrm{level}(\tau), \mathrm{level}(\tau'))$.)

First we extend the type structure with a base type $\triangle$ for tree ordinals.

5.1. Definition. Types are 0 and $\triangle$, and with τ, τ' also $\tau{\to}\tau'$. Furthermore, define $\mathcal{N}_0 = \mathcal{N}$, $\mathcal{N}_\triangle = \mathrm{TO}_0$ and $\mathcal{N}_{\tau\to\tau'} = \mathcal{N}_\tau{\to}\mathcal{N}_{\tau'}$, the set of all mappings from $\mathcal{N}_\tau$ to $\mathcal{N}_{\tau'}$. We call $\bigcup\mathcal{N}_\tau$ (with τ ranging over the extended types) the full typestructure over $\mathcal{N}$ and TO_0. Application, notation etcetera are similar to those for $\mathcal{M}$ given in 2.1.

In view of the encoding of tree ordinals as type 2 functionals, the following embedding of the extended types in the types of Section 2 (i.e. with 0 as the only base type) should not come as a surprise.

5.2. Definition. $\mathrm{t}(0) = 0$; $\mathrm{t}(\triangle) = (0{\to}0){\to}0$; $\mathrm{t}(\tau{\to}\tau') = \mathrm{t}(\tau){\to}\mathrm{t}(\tau')$.

In 2.2 we defined BR, the bar recursive functionals in $\mathcal{M}$. Here we define TR, the *tree recursive* functionals in $\bigcup\mathcal{N}_\tau$. We essentially follow Zucker [15], where this system of functionals is called T_1 (trees of the *first* order; Zucker also considers T_2, trees of the *second* order, i.e. T_1-branching trees). A system similar to T_1, but somewhat more involved, has been defined by Howard [5] (the theory V).

5.3. Definition. Consider the following basic functionals: $0_\triangle = \underline{0} \in \mathcal{N}_\triangle$; $\boldsymbol{S}_\triangle = \lambda a.\, <a> \in \mathcal{N}_\triangle{\to}\mathcal{N}_\triangle$; $\boldsymbol{J}_\triangle = \lambda f.\, \ll fn\gg_{n\in\mathcal{N}} \in (\mathcal{N}_0{\to}\mathcal{N}_\triangle){\to}\mathcal{N}_\triangle$; $\boldsymbol{R}_\triangle$ defined by:

$$\boldsymbol{R}_\triangle ZSJ0_\triangle = Z,\ \boldsymbol{R}_\triangle ZSJ<a> = S(\boldsymbol{R}_\triangle ZSJa) \text{ and}$$
$$\boldsymbol{R}_\triangle ZSJ\ll a_n\gg = J(\lambda n.\, \boldsymbol{R}_\triangle ZSJa_n)\ll a_n\gg$$

The type of $\boldsymbol{R}_\triangle$ is determined by the type of Z (but suppressed in the denotation $\boldsymbol{R}_\triangle$). These functionals are zero, successor, join and primitive recursor for tree ordinals, respectively, to be compared with the basic functionals for primitive recursion: 0, $\boldsymbol{S}$ and $\boldsymbol{R}$. A *tree recursive functional* is a functional that can be constructed from 0, $\boldsymbol{S}$, primitive recursors, $0_\triangle$, $\boldsymbol{S}_\triangle$, $\boldsymbol{J}_\triangle$ and tree recursors, using lambda abstraction and application. Let TR (TR_τ) denote the set of tree recursive functionals (of type τ).

We are interested in representing functionals of TR with functionals of BR. Given the embedding of the types, we define, for every extended type τ, when a functional from $\mathcal{M}_{\mathrm{t}(\tau)}$ represents a functional from $\mathcal{N}_\tau$.

5.4. Definition. By induction on τ we define relations $\rightsquigarrow_\tau \subset \mathcal{M}_{\mathrm{t}(\tau)} \times \mathcal{N}_\tau$ as follows:

$$\begin{array}{rcl} m \rightsquigarrow_0 n & \Leftrightarrow & m = n \\ Y \rightsquigarrow_\triangle a & \Leftrightarrow & Y = Y_a \\ A \rightsquigarrow_{\tau\to\tau'} A' & \Leftrightarrow & \forall B \in \mathcal{M}_{\mathrm{t}(\tau)} \forall B' \in \mathcal{N}_\tau (B \rightsquigarrow_\tau B' \Rightarrow AB \rightsquigarrow_{\tau'} A'B') \end{array}$$

Note that this definition is a proper generalization of Definition 4.6, with $f : \mathrm{TO}_0 \to \mathrm{TO}_0$ replaced by $A' \in \mathcal{N}_{\triangle\to\triangle}$. From now on we shall omit type subscripts and simply write $X \rightsquigarrow X'$ whenever X represents X'.

5.5. Theorem. *Every functional from* TR *can be represented by a functional from* BR.

Proof. We shall define a mapping $\mathrm{t} : \mathrm{TR} \to \mathrm{BR}$ and prove that $\mathrm{t}(X) \rightsquigarrow X$ for all $X \in \mathrm{TR}$. Both will be done by induction on X. We first treat the inductive cases and then the basic functionals. Application presents no difficulty: since BR (like TR) is closed under application we can simply take $\mathrm{t}(AB) = \mathrm{t}(A)\mathrm{t}(B)$ and obtain $\mathrm{t}(AB) \rightsquigarrow AB$ by the induction hypotheses $\mathrm{t}(A) \rightsquigarrow A$ and $\mathrm{t}(B) \rightsquigarrow B$ and Definition 5.4. Lambda abstraction is somewhat more involved. Instead of extending our notion $\rightsquigarrow$ to functionals containing variables, we prefer to proceed as follows: (i) lambda abstraction can be defined using the well-known combinators $\Pi_{\tau,\tau'} = \lambda XY.\, X \in \mathcal{N}_{\tau\to\tau'\to\tau}$ and $\Sigma_{\tau,\tau',\tau''} = \lambda XYZ.\, XZ(YZ) \in \mathcal{N}_{\ldots}$; (ii) $\Pi_{\tau,\tau'}$ and $\Sigma_{\tau,\tau',\tau''}$ can be represented by $\Pi_{\mathrm{t}(\tau),\mathrm{t}(\tau')} \in \mathrm{BR}$ and $\Sigma_{\mathrm{t}(\tau),\mathrm{t}(\tau'),\mathrm{t}(\tau'')} \in \mathrm{BR}$ (the proof of this fact uses Definition 5.4 a number of times). It remains to show that the basic functionals of TR can be represented. All but one are straightforward. For example, $\boldsymbol{J}_\triangle$ is represented by J from the proof of Lemma 4.5 and $\boldsymbol{S}_\triangle$ is represented by $S_{<.>}$ defined just after Definition 4.6. The only difficult one is $\boldsymbol{R}_\triangle$, since it involves a generalization of the functional $\boldsymbol{I}$ from 4.2. Define

$$\begin{array}{rcl} \boldsymbol{R}^-_\triangle & = & \lambda ZSJY.\, \boldsymbol{B}Y G_Z H_{Y,S,J}, \text{ where} \\ G_Z & = & \lambda\sigma.\, Z \text{ and} \\ H_{Y,S,J} & = & \lambda A\sigma.\ \mathsf{if}\ \ \mathrm{Succ}(Y,\sigma)\ \ \mathsf{then}\ \ S(A0) \\ & & \quad\ \mathsf{elsif}\ \ \mathrm{Join}(Y,\sigma)\ \ \mathsf{then}\ \ JA(\lambda F.\, Y(\sigma * F)) \\ & & \quad\ \mathsf{else}\ \ 0 \end{array}$$

Here $\sigma * F$ stands for the function assigning σ_i to i if $i < \mathrm{lh}(\sigma)$, and $F(\mathrm{lh}(\sigma) - i)$ otherwise. We have $\lambda F.\, Y_a(\sigma * F) = Y_{a\downarrow\sigma}$ for all $\sigma \in \mathrm{Ortree}(Y)$ by the results of Section 3. The only difference between $\boldsymbol{I}$ and $\boldsymbol{R}^-_\triangle$ is the argument $\lambda F.\, Y(\sigma * F)$ of JA, which is a consequence of the argument $\ll a_n \gg$ in the third clause of the definition of $\boldsymbol{R}_\triangle$ in 5.3 (this makes the difference between a *recursor* and an *iterator*). Now assume $Z \rightsquigarrow Z'$, $S \rightsquigarrow S'$ and $J \rightsquigarrow J'$. By induction on $a\downarrow\sigma$ (using Lemma 3.8) one proves $\boldsymbol{R}^-_\triangle ZSJY_a\sigma = \boldsymbol{R}_\triangle Z'S'J'(a\downarrow\sigma)$ for all $\sigma \in \mathrm{Ortree}(Y_a)$. Now put $\mathrm{t}(\boldsymbol{R}_\triangle) = \lambda ZSJY.\, \boldsymbol{R}^-_\triangle ZSJY{<>}$. It follows by iterated application of Definition 5.4 that $\mathrm{t}(\boldsymbol{R}_\triangle) \rightsquigarrow \boldsymbol{R}_\triangle$. $\square$

5.6. Corollary. *For every* $a \in \mathrm{TR}_\triangle$ *we have* $Y_a \in \mathrm{BR}$.

5.7. Corollary. $|\mathrm{ID}_1| \leq \sup\{|a| \mid a \in \mathrm{TO}_0 \wedge Y_a \in \mathrm{BR}\}$.

The latter corollary is a simplified version of Vogel [14, Satz b] combined with Zucker [15], where the ordinal of ID_1 is proved to be equal to the ordinal of T_1 (= TR). Other characterizations of this same ordinal are: η_0, the Howard ordinal, and $\phi\epsilon_{\Omega+1}0$ in the Bachmann hierarchy. See also Terlouw [13] for a detailed ordinal analysis of ID_1.

One can imagine that trees of the second order, i.e. T_1-branching trees, can be encoded by functionals of type $(0\to2)\to0$, and so on. This is possible indeed. However, $|\mathrm{ID}_\omega|$ is known to be much weaker than full analysis and hence these encodings are certainly not exhausting the full strength of bar recursion. Further generalizations should be possible but will not be pursued here. A related question is whether $\leq$ in Corollary 5.7 is $=$ or $<$.

6. Metamathematical considerations

One could ask in how far bar recursion is essential for the construction of the functional Q from Lemma 4.5. For metamathematical reasons, to be made precise below, the functionals $Y_{\underline{\varepsilon}_0}$, $\boldsymbol{I}Y_{\underline{\varepsilon}_0}$ (a transfinite $\underline{\varepsilon}_0$ iterator) and $\boldsymbol{B}Y_{\underline{\varepsilon}_0}$ (a transfinite $\underline{\varepsilon}_0$ recursor) surpass the strength of higher order primitive recursion. As a consequence, the construction of $Y_{\underline{\omega}^a}$ from Y_a, uniformly for all $a \in \mathrm{TO}_0$, cannot be done by primitive recursion. When one nevertheless tries to do this, one sees that already for a with $|a| < \varepsilon_0$ primitive recursors of arbitrarily high types (depending on $|a|$) are needed. A very clear example of this phenomenon is given by Fortune, Leivant and O'Donnell [2], where the dependency of terms on types (polymorphism) in $\lambda2$ is exploited to circumvent this problem. In Section 4 we constructed a functional representing $\lambda a.\, \underline{\omega}^a$ by exploiting directly the dependency of terms on *order* types. So the answer to the above question is that $\lambda2$ provides an alternative to the bar recursive construction of Q, but that primitive recursion is definitely too weak.

In this section we relate our results to [2] and we substantiate the metamathematical reasons mentioned above.

In [2] base ω Cantor normal forms are used: $\mathrm{Cnf}(\alpha) \equiv \omega^{\alpha_1} + \cdots + \omega^{\alpha_k} = \alpha$ with $\alpha \geq \alpha_1 \geq \ldots \geq \alpha_k$ $(k \geq 0)$; $\mathrm{Cnf}(0)$ is the empty sum. Instead of using coefficients we write the appropriate sum of powers of ω. For the rest of this paper, ordinals will be at most ε_0, unless explicitly stated otherwise.

Limit ordinals $\alpha < \varepsilon_0$ are approximated by so-called fundamental sequences $\alpha[n]$ $(n \in \mathcal{N})$ in a well-known way: if $\mathrm{Cnf}(\alpha) \equiv \omega^{\alpha_1} + \cdots + \omega^{\alpha_k} < \varepsilon_0$ with $\alpha_k > 0$ then

$$\alpha[n] = \begin{cases} \omega^{\alpha_1} + \cdots + \omega^{\alpha_k - 1} \cdot n & \text{if } \mathrm{Succ}(\alpha_k) \\ \omega^{\alpha_1} + \cdots + \omega^{\alpha_k[n]} & \text{if } \mathrm{Lim}(\alpha_k) \end{cases}$$

Note that this inductive definition is correct since $\alpha < \varepsilon_0$ implies $\alpha_k < \alpha$. The ordinal ε_0 is approximated by $\varepsilon_0[n] = (\lambda\alpha.\, \omega^\alpha)^n 0$.

Based on the approximation of limit ordinals defined above, the following slightly different notion of transfinite iteration is used in [2].

6.1. Definition. For all $\alpha \leq \varepsilon_0$ the α-iteration denoted by $(J;S)^\alpha Z$ is defined by:

$$\begin{array}{rcll} (J;S)^0 Z & = & Z & \\ (J;S)^{\alpha+1} Z & = & S((J;S)^\alpha Z) & \\ (J;S)^\alpha Z & = & J(\lambda n.\, (J;S)^{\alpha[n]} Z) & \text{when } \mathrm{Lim}(\alpha) \end{array}$$

Mind the difference between $(J;S)^a Z$ and $(J;S)^\alpha Z$. We are interested in cases in which the equation $(J;S)^a Z = (J;S)^{|a|} Z$ holds. However, for $\ll a_n \gg$ with $a_n = \underline{0}$ for all $n \in \mathcal{N}$ we have $|\ll a_n \gg| = 1$ and the equation does not hold since 1 is not a limit ordinal. Moreover, for $\ll a_n \gg$ with $a_n = \underline{n+1}$ for all $n \in \mathcal{N}$ we have $|\ll a_n \gg| = \omega$ but the equation does not hold since $|a_n| = n+1$ and $|\ll a_n \gg|[n] = n$. These simple counterexamples lead to the following canonical embedding of ordinals $\alpha < \varepsilon_0$ in TO_0.

6.2. Definition. For $\alpha \leq \varepsilon_0$ we define $\triangle(\alpha) \in \mathrm{TO}_0$ by induction as follows: $\triangle(0) = \underline{0}$; $\triangle(\alpha+1) = <\triangle(\alpha)>$; $\triangle(\alpha) = \ll \triangle(\alpha[n]) \gg$ when $\mathrm{Lim}(\alpha)$.

We have the following lemmas.

6.3. Lemma. *For all $\alpha \leq \varepsilon_0$ we have $|\triangle(\alpha)| = \alpha$.*

Proof. By a trivial induction on α. □

6.4. Lemma. *If $\mathrm{Lim}(\beta)$ and $\beta < \omega^{\alpha+1} < \varepsilon_0$, then we have $(\omega^\alpha + \beta)[n] = \omega^\alpha + (\beta[n])$.*

Proof. Obvious. □

6.5. Lemma. *For all α, β we have:*

(i) *if $\beta < \omega^{\alpha+1} < \varepsilon_0$, then $\triangle(\omega^\alpha + \beta) = \triangle(\omega^\alpha) + \triangle(\beta)$*

(ii) *if $\alpha < \varepsilon_0$, then $\triangle(\omega^\alpha) = \underline{\omega}^{\triangle(\alpha)}$*

(iii) $\triangle(\varepsilon_0) = \underline{\varepsilon}_0$

Proof. (i) By induction on β using Lemma 6.4. (ii) By induction on α using (i). (iii) Recall that $\underline{\varepsilon}_0 = \ll (\lambda a.\, \underline{\omega}^a)^n(\underline{0}) \gg$. From (ii) it follows by induction on n that $(\lambda a.\, \underline{\omega}^a)^n(\underline{0}) = \triangle(\varepsilon_0[n])$, and hence $\triangle(\varepsilon_0) = \underline{\varepsilon}_0$. □

6.6. Lemma. *For all $\alpha \leq \varepsilon_0$ we have:*

$$(J;S)^{\triangle(\alpha)} Z = (J;S)^\alpha Z$$

PROOF. By induction on α. $\square$

6.7. COROLLARY. $(J;S)^{\underline{\varepsilon}_0}Z = (J;S)^{\varepsilon_0}Z$

6.8. THEOREM. *There is no primitive recursive functional representing* $\lambda a.\,\underline{\omega}^a$.

PROOF. Consider the function (functional of type 1) defined by $F_{\varepsilon_0} = (J;S)^{\varepsilon_0}Z$ with

$$\begin{aligned} Z &= \lambda n.\, n+1, \\ S &= \lambda Fn.\, F^n n, \\ J &= \lambda Xy.\, Xyy \qquad \text{(diagonalizes in the limit case).} \end{aligned}$$

This function is also used in [2] and has been attributed in Schwichtenberg [10] to Robbin [9] (actually with $Z = \lambda n.\, 2^n$, but for the purpose here it suffices that Z is strictly increasing). As stated in [10, p. 61, l. -7], F_{ε_0} is not provably total recursive in Peano Arithmetic. Now we argue as follows:

F_{ε_0} is not provably total recursive in PA	$\Rightarrow$	(by [4], [7, §3.4])
$(J;S)^{\varepsilon_0}Z$ is not primitive recursive	$\Rightarrow$	(by Cor. 6.7)
$(J;S)^{\underline{\varepsilon}_0}Z$ is not primitive recursive	$\Rightarrow$	(by Th. 4.3)
$\boldsymbol{I}Y_{\underline{\varepsilon}_0}$ is not primitive recursive	$\Rightarrow$	(by Def. 4.2)
$\boldsymbol{B}Y_{\underline{\varepsilon}_0}$ is not primitive recursive	$\Rightarrow$	(by [11])
$Y_{\underline{\varepsilon}_0}$ is not primitive recursive	$\Rightarrow$	(by Th. 4.7)
Q is not primitive recursive		

Here Q is any functional of type $2\to 2$ representing $\lambda a.\,\underline{\omega}^a$, in particular the bar recursive functional Q from Lemma 4.5. $\square$

Acknowledgement

The research in this paper has been supported by the Netherlands Computer Science Research Foundation (SION) with financial support of the Netherlands Organization for Scientific Research (NWO).

References

[1] M.A. Bezem. Strongly majorizable functionals of finite type: a model for bar recursion containing discontinuous functionals. *Journal of Symbolic Logic*, 50:652–660, 1985.

[2] S. Fortune, D. Leivant, and M. O'Donnell. The expressiveness of simple and second-order type structures. *Journal of the ACM*, 30:151–185, 1983.

[3] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

[4] K. Gödel. Ueber eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287, 1958.

[5] W.A. Howard. A system of abstract constructive ordinals. *Journal of Symbolic Logic*, 37:355–374, 1972.

[6] W.A. Howard. Ordinal analysis of simple cases of bar recursion. *Journal of Symbolic Logic*, 46:17–30, 1981.

[7] G. Kreisel. Interpretation of analysis by means of constructive functionals of finite types. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland, 1959.

[8] W. Pohlers. *Proof Theory*. Number 1407 in Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1989.

[9] J.W. Robbin. *Subrecursive hierarchies*. PhD thesis, Princeton, 1965.

[10] H. Schwichtenberg. Eine Klassifikation der ε_0-rekursiven Funktionen. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 17:61–74, 1971.

[11] H. Schwichtenberg. On bar recursion of types 0 and 1. *Journal of Symbolic Logic*, 44:325–329, 1979.

[12] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In J.C.E. Dekker, editor, *Recursive function theory, Proc. Symp. in pure mathematics V*, pages 1–27. AMS, Providence, 1962.

[13] J. Terlouw. *Proof-theoretical analyses of transfinite recursion and inductive definitions*. PhD thesis, Utrecht University, 1986.

[14] H. Vogel. Ueber die mit Bar Rekursion vom Type 0 definierbaren Ordinalzahlen. *Archive for Mathematical Logic*, 19:165–173, 1978.

[15] J.I. Zucker. Iterated inductive definitions, trees and ordinals. In A.S. Troelstra, editor, *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, pages 392–453. Springer-Verlag, Berlin, 1973.