Arch. Math. Logic (1988) 27: 149–160

Archive for Mathematical Logic
© Springer-Verlag 1988

Equivalence of Bar Recursors in the Theory of Functionals of Finite Type

Marc Bezem

Centre for Mathematics and Computer Science, P.O. Box 4079, NL-1009 AB Amsterdam, The Netherlands

Abstract. The main result of this paper is the equivalence of several definition schemas of bar recursion occurring in the literature on functionals of finite type. We present the theory of functionals of finite type, in [T] denoted by $qf-WE-\mathbf{HA}^\omega$, which is necessary for giving the equivalence proofs. Moreover we prove two results on this theory that cannot be found in the literature, namely the deduction theorem and a derivation of Spector's rule of extensionality from [S]: if $P \to T_1 = T_2$ and $Q[X :\equiv T_1]$, then $P \to Q[X :\equiv T_2]$, from the at first sight weaker rule obtained by omitting "$P\to$".

Chapter 1. Introduction to Language and Theory of Functionals of Finite Type

§ 1. The Language

1.1. *Types* are 0 and with σ and τ also $(\sigma)\tau$ (often written as $\sigma \to \tau$ in the literature).

The language for functionals of finite type is quantifier-free, contains the propositional operators $\wedge$, $\vee$, $\to$, and $\neg$, variables for each type, constants of types specified below, and equality $=$ between objects of type 0. For the *variables* we use $a^0, \ldots, z^0, A^\sigma, \ldots, Z^\sigma$ for all types σ; type superscripts are often omitted. The *constants* are ($\in$ expresses "of type")

$$0 \in 0 \quad \text{(zero)},$$

$$\$ \in (0)0 \quad \text{(successor)},$$

$$K_{\sigma,\tau} \in (\sigma)(\tau)\sigma \quad \text{(combinator } K\text{)},$$

$$S_{\varrho,\sigma,\tau} \in ((\varrho)(\sigma)\tau)((\varrho)\sigma)(\varrho)\tau \quad \text{(combinator S)},$$

$$R_\sigma \in (\sigma)((\sigma)(0)\sigma)(0)\sigma \quad \text{(primitive recursor)},$$

$$B_{\sigma,\tau} \in (((0)\sigma)0)(((0)\sigma)(0)\tau)(((\sigma)\tau)((0)\sigma)(0)\tau)((0)\sigma)(0)\tau \quad \text{(bar recursor)},$$

for all types ϱ, σ, τ; type subscripts are mostly omitted. Confusion of variables and constants (introduced by omitting super- and subscripts) will be avoided by not using the letters K, S, R and B for variables.

The set of *terms* of the theory $\mathbf{T} \cup BR$ contains all variables and constants and is closed under the following *application*: if T_1 is a term of type $(\sigma)\tau$ and T_2 is a term of type σ, then $(T_1 T_2)$ is a term of type τ. Thus for application no other symbols than (and) are needed, since application is denoted by juxtaposition. We further reduce the notational overhead by taking association to be to the left and by omitting outer parentheses. We use t resp. T as syntactical variables for terms of type 0 ($t \in 0$) resp. terms of type σ ($T \in \sigma$). Syntactical identity of terms will be expressed by $\equiv$. *Numerals* are defined by $\underline{0} \equiv 0$; $\underline{n+1} \equiv \$\underline{n}$ (n is here used as a meta-variable). Underlining will be omitted when confusion is not likely. *Closed terms* (or *functionals*) are terms not containing variables. We use F as syntactical variable for closed terms of type σ ($F \in \sigma$).

Prime formulas are equations between terms of type 0. *Formulas* are constructed from prime formulas with the help of the propositional operators. As syntactical variables for formulas we use P and Q. Substitution, e.g. of a term T for *all* occurrences of a variable X in a formula P, will be denoted by $P[X :\equiv T]$.

Confusion of variables with arbitrary terms, numerals, closed terms or formulas will be avoided by not using t, T, n, F, P and Q for variables.

1.2. Let us sum up the notational conventions introduced above.

$0, \$, K, S, R, B$	constants
$t, t_1, t_2, \ldots$	terms of type 0
$T, T_1, T_2, \ldots$	terms of arbitrary type σ
$n, \underline{n}, 0, 1, \ldots$	numerals
$F, F_1, F_2, \ldots$	closed terms
$P, Q, Q[y :\equiv t], \ldots$	formulas
all other lower case letters	variables of type 0
all other capitals	variables of arbitrary type σ.

§ 2. The Theory

2.1. Gödels theory $\mathbf{T}$ of primitive recursive functionals is axiomatized as a Hilbert-type system. Derivations do not depend on assumptions. Axioms and rules of inference are specified below.

For all terms T_1, T_2 of type $(\sigma_1)\ldots(\sigma_k)0$, let $T_1 = T_2$ be a permanent abbreviation of $T_1 X^{\sigma_1} \ldots X^{\sigma_k} = T_2 X^{\sigma_1} \ldots X^{\sigma_k}$, where $X^{\sigma_1}, \ldots, X^{\sigma_k}$ are distinct variables not occurring in T_1, T_2.

Axioms and Rules of $\mathbf{T}$:

Rules and axioms of intuitionistic propositional calculus, e.g. as in [T, 1.1.3]. A substitution rule:

$$\text{if } P, \text{ then } \quad P[X^\sigma :\equiv T] \quad (T \in \sigma).$$

Equality axioms:

$$x=x, \quad (x=y\wedge x=z)\to y=z, \quad x=y\to t[z:\equiv x]=t[z:\equiv y].$$

Successor axioms:

$$\neg\, 0=\$x, \quad \$x=\$y\to x=y.$$

A rule of induction:

$$\text{if}\quad P[x:\equiv 0] \quad\text{and}\quad P\to P[x:\equiv \$x], \quad\text{then } P.$$

The following defining equations for the constants (of all appropriate types):

$$KXY=X, \quad SXYZ=XZ(YZ), \quad RXY0=X, \quad RXY(\$z)=Y(RXYz)z.$$

A rule of extensionality:

$$\text{if}\quad P\to T_1=T_2 \quad\text{and}\quad Q[X:\equiv T_1], \quad\text{then}\quad P\to Q[X:\equiv T_2]$$

(provided that the variables that are suppressed in the abbreviation $T_1=T_2$ occur neither in P, nor in Q).

This completes the description of **T**. In the terminology of [T] this theory is one of the theories called $qf-WE-\mathbf{HA}^{\omega}$ (the extensionality rule may vary a little, see [T, 1.6.12ff.]).

2.2. Before we can give the defining equations for the constant B, the bar recursor, we have to make two extra provisions.

Firstly we apply Curry's method from [CF] to define λ-abstraction. By induction on the construction of terms we define $\lambda X\cdot X\equiv SKK$, $\lambda X\cdot T\equiv KT$ (T a constant or a variable different from X) and $\lambda X\cdot T_1T_2\equiv S(\lambda X\cdot T_1)(\lambda X\cdot T_2)$. Thus for every term T there exists a term $\lambda X\cdot T$ such that $(\lambda X\cdot T)Y=T[X:\equiv Y]$.

Secondly we need some special primitive recursive functionals. Define constant functionals of type σ by

$$n^0\equiv \underline{n};$$

$$n^{(\sigma)\tau}\equiv K_{\tau,\sigma}n^{\tau} \quad \text{for all } n.$$

In order to avoid confusion with numerals we shall not omit type superscripts in denotations of constant functionals. For all types σ there exist primitive recursive functionals $[\,]$ and $*$ such that (cf. [L, p. 22])

$$y<x\to[C]_x y=Cy, \quad y\geqq x\to[C]_x y=0^{\sigma},$$

and

$$y\neq x\to(C*_x X)y=Cy, \quad y=x\to(C*_x X)y=X$$

are provable in **T**. Here C is of type $(0)\sigma$, X of type σ, and $<$, $\geqq$, $\neq$ abbreviate their codifications in **T**. Moreover we shall write $x+1$ for $\$x$.

Bar recursion is a principle of definition by recursion on a well-founded tree of finite sequences of functionals of type σ. Following Spector [S] we use the pair $([C]_x, x)$ to represent the finite sequence $\langle C0,\ldots,C(x-1)\rangle$. The *defining equations for B* (of all appropriate types) are:

$$Y[C]_x<x\to BYGHCx=G[C]_x x,$$

$$Y[C]_x\geqq x\to BYGHCx=H(\lambda X\cdot BYGH[C*_x X]_{x+1}(x+1))[C]_x x.$$

These defining equations are often referred to by "*the schema of bar recursion*" or "*the definition schema of B*" and are written informally (omitting Y, G, H as arguments of B) as

$$BCx = \begin{cases} G[C]_x x & \text{if } \; Y[C]_x < x, \\ H(\lambda X \cdot B[C *_x X]_{x+1}(x+1))\,[C]_x x & \text{else}. \end{cases}$$

Let $BR_{\sigma,\tau}$ denote the definition schema of $B_{\sigma,\tau}$ and let $BR_\sigma = \cup_\tau BR_{\sigma,\tau}$, $BR = \cup_\sigma BR_\sigma$. $\mathbf{T} \cup BR$ is thus simply $\mathbf{T}$ with axioms BR added.

§ 3. *Remarks on the Theory*

3.1. Without excessive effort (see [L, p. 20]) the decidability of prime formulas can be shown in $\mathbf{T}$ and $\mathbf{T} \cup BR$, i.e.,

$$\vdash x = y \vee x \neq y .$$

Since $\mathbf{T} \cup BR$ does only contain propositional operators and no quantifiers, it follows by formula induction that all formulas are decidable. As a consequence we could have taken classical instead of intuitionistic propositional logic. However, we opted for intuitionistic logic, so that $\mathbf{T} \cup BR$ is a member of the family $\ldots - \mathbf{HA}^\omega + \ldots$ described in [T].

Another consequence is that $\mathbf{T} \cup BR$ can be presented as an equational calculus. For, the classical truth functions are primitive recursive and can hence be represented by certain functionals in $\mathbf{T}$. Replacing the propositional connectives by these functionals changes every formula into an equivalent formula of the form $t = 0$.

3.2. As stated in Sect. 2, we only consider derivations without assumptions in $\mathbf{T} \cup BR$ ("$\vdash$"). This is considered no restriction, provided that the deduction theorem holds in case also derivations depending on assumptions are allowed ("$\Gamma \vdash$"). However, liberalizing rules of inference from "$\vdash$" to "$\Gamma \vdash$" must be done carefully. Of course the substitution rule (if P, then $P[X :\equiv T]$) and the induction rule (if $P[x :\equiv 0]$ and $P \to P[x :\equiv \$x]$, then P) only remain correct if X and x do not occur in any assumption on which the premiss depends. The same applies to the variables that are suppressed by the abbreviation $T_1 = T_2$ in the rule of extensionality (if $P \to T_1 = T_2$ and $Q[X :\equiv T_1]$, then $P \to Q[X :\equiv T_2]$). All other rules are unproblematic.

Now the deduction theorem is proved as usual by induction on the length of derivations. The only step in this proof that we did not find in the literature, although not very different from the other steps, is the following. Suppose the last step in some derivation with assumptions $P_0, P_1, \ldots, P_n$ is an inference by the rule of extensionality:

$$\text{if} \quad P \to T_1 = T_2 \quad \text{and} \quad Q[X :\equiv T_1], \quad \text{then} \quad P \to Q[X :\equiv T_2] .$$

Then the variables suppressed by the abbreviation $T_1 = T_2$ do not occur in P, Q, nor in any assumption on which the premiss depends. By the induction hypothesis from the proof of the deduction theorem we have

$$P_1, \ldots, P_n \vdash P_0 \to (P \to T_1 = T_2), \qquad P_0 \to Q[X :\equiv T_1] .$$

Hence by intuitionistic propositional logic

$$P_1, \ldots, P_n \vdash (P_0 \wedge P) \rightarrow T_1 = T_2, \qquad P_0 \rightarrow Q[X :\equiv T_1].$$

Since variable X has only a syntactical meaning in $Q[X :\equiv T_i]$, it can be renamed. So we can assume without loss of generality that X does not occur in P_0. Hence $(P_0 \rightarrow Q)[X :\equiv T_i]$ is identical to $P_0 \rightarrow Q[X :\equiv T_i]$ $(i = 1, 2)$. Moreover, no variables occur anywhere they should not, hence by the rule of extensionality we have

$$P_1, \ldots, P_n \vdash (P_0 \wedge P) \rightarrow (P_0 \rightarrow Q[X :\equiv T_2]).$$

By intuitionistic propositional logic we can conclude

$$P_1, \ldots, P_n \vdash P_0 \rightarrow (P \rightarrow Q[X :\equiv T_2]). \quad \square$$

3.3. One might ask for the reason of the double use of "$P\rightarrow$" in the extensionality rule

$$\text{if} \quad P \rightarrow T_1 = T_2 \quad \text{and} \quad Q[X :\equiv T_1], \quad \text{then} \quad P \rightarrow Q[X :\equiv T_2].$$

Taking $0 = 0$ for P yields the rule

$$\text{if} \quad T_1 = T_2 \quad \text{and} \quad Q[X :\equiv T_1], \quad \text{then} \quad Q[X :\equiv T_2].$$

The reason is simply that the defining equations of the bar recursor are of the form $P \rightarrow T_1 = T_2$. The rule with "$P\rightarrow$" is used (implicitly) in [S] in the proof of the soundness of the Dialectica interpretation. Moreover, the part of the proof of the deduction theorem given above fails for the rule without "$P\rightarrow$". In the literature we could not find any remark to the effect that the rule with "$P\rightarrow$" can be derived from the rule without "$P\rightarrow$". We therefore prove the following

Lemma. *The rule of extensionality with "$P\rightarrow$" can be derived from the rule of extensionality without "$P\rightarrow$".*

Proof. Assume the rule without "$P\rightarrow$". We shall prove the rule with "$P\rightarrow$" with $t_1 = 0$ for P and $t = 0$ for Q. By 3.1 this is sufficient for the rule with "$P\rightarrow$". Suppose $t_1 = 0 \rightarrow T_1 = T_2$ and $t[X :\equiv T_1] = 0$ have been derived. Define (for $i = 1, 2$)

$$T_i' \equiv R T_i 0^{(\sigma)(0)\sigma}, \quad \text{with } \sigma \text{ the type of } T_i.$$

Then we have (for $i = 1, 2$ and x not occurring in T_i):

$$(*) \qquad \begin{cases} x = 0 \rightarrow T_i' x = T_i, \\ x \neq 0 \rightarrow T_i' x = 0^\sigma. \end{cases}$$

From $T_i' 0 = T_i$ it follows by extensionality without "$P\rightarrow$" that $t[X :\equiv T_i] = t[X :\equiv T_i' 0]$. By the axiom $x = y \rightarrow t[z :\equiv x] = t[z :\equiv y]$ we have $t_1 = 0 \rightarrow t[X :\equiv T_i' 0] = t[X :\equiv T_i' t_1]$. By $t_1 = 0 \rightarrow T_1 = T_2$ (and the decidability of $t_1 = 0$) it follows from $(*)$ that $T_1' t_1 = T_2' t_1$, and hence we have, again by extensionality without "$P\rightarrow$", $t[X :\equiv T_1' t_1] = t[X :\equiv T_2' t_1]$. It follows that

$$\begin{aligned} t_1 = 0 \rightarrow t[X :\equiv T_2] &= t[X :\equiv T_2' 0] = t[X :\equiv T_2' t_1] \\ &= t[X :\equiv T_1' t_1] = t[X :\equiv T_1' 0] = t[X :\equiv T_1] = 0. \quad \square \end{aligned}$$

As a consequence we obtain the deduction theorem for the rule without "$P\to$" (the disproof of this fact in [T, 1.6.12] is not correct, since $x^\sigma = y^\sigma$ is an assumption containing suppressed variables, and as such no legal premiss of the rule of extensionality). We are indebted to Henk Barendregt for his persistence in urging us to prove (or disprove) the equivalence of both extensionality rules.

3.4. It is worth noting that pairing is possible in $\mathbf{T}\cup BR$. As a consequence we can reduce the study of simultaneous recursors etc. to the single case.

3.5. We could have taken λ-abstraction as a primitive, instead of the combinatorial version of Sect. 2. Because of the presence of the extensionality rule, these two versions are equivalent (in the sense of [Ba, 7.3.10]).

Chapter 2. Some Equivalent Bar Recursors

§ 1. Introduction

1.1. Let [X] and [Y] be any of [S, H, Ta, V] or [B]. Let $B^X_{\sigma,\tau}$ be the bar recursor occurring in place [$X\cdot$] in literature, with defining equations $BR^X_{\sigma,\tau}$. We shall construct, primitive recursively in $B^Y_{\sigma,\tau'}$ for suitable type τ' depending on σ and τ, a functional B which satisfies, provably in $\mathbf{T}\cup BR^Y$, the defining equations of $B^X_{\sigma,\tau}$, and vice versa. Hence $\mathbf{T}\cup BR^X$ and $\mathbf{T}\cup BR^Y$ are equivalent.

1.2. Bar recursors can differ in many respects, such as:
(1) permutations of arguments
(2) different representations of finite sequences of functionals
(3) use of λ-operator or λ-free
(4) number of arguments of G and H.
We consider (1) as trivial and shall only pay attention to differences of the kinds (2), (3), and (4).

1.3. Though the proofs are presented informally, they can easily be formalized in $\mathbf{T}\cup BR$ ("by induction" and "by extensionality" refer to the corresponding rules of $\mathbf{T}$). The general form of all proofs in this chapter is the following: Assume $P\to T_1 = T_2$ by BR^Y. By induction and extensionality we prove $P'\leftrightarrow P$, $T_1 = T_1'$, $T_2 = T_2'$, and hence $P'\to T_1' = T_2'$. It follows that BR^X holds.
Extensionality will only be used in the form without "$P\to$" (see Chap. 1, 3.3).

1.4. We shall slightly deviate from the notational conventions introduced in the previous chapter. Bar recursors that are constants are denoted by B^S, B^V etc., whereas B is used to denote a defined bar recursor. Moreover, Y^S, G^H, H^T etc. abbreviate certain defined terms. We shall often suppress the denotations of the first three arguments of a bar recursor. This will be indicated by underlining the denotation of the bar recursor, e.g., $\underline{B}$ abbreviates $BYGH$.

§ 2. Spector$\leftrightarrow$Howard

2.1. We recall from Chap. 1 Spector's schema of bar recursion:

$$(BR^S) \qquad \underline{B}^S Cx = \begin{cases} G[C]_x x & \text{if } Y[C]_x < x, \\ H(\lambda X\cdot \underline{B}^S[C *_x X]_{x+1}(x+1))[C]_x x & \text{else}. \end{cases}$$

Howard formulates the schema of bar recursion directly in terms of finite sequences of functionals. Therefore we have to extend language and theory with types $\sigma^{\cup}$ for finite sequences $\langle\cdot,\dots,\cdot\rangle$ of functionals of type σ, with variables $\alpha^{\sigma^{\cup}}$ for each type $\sigma^{\cup}$, as well as length functionals lh, concatenation operators $*$ (addition of one element at the end) and projection operations: α_i equals the i-th component of α if $i<lh\,\alpha$, and 0^{σ} else. These objects are given their usual properties by suitable axioms. Let $[\alpha]=\lambda i\cdot\alpha_i\in(0)\sigma$ and $\bar{C}x=\langle C0,\dots,C(x-1)\rangle\in\sigma^{\cup}$ for $C\in(0)\sigma$. Note that the extension is definitional, since finite sequences of functionals can be coded, e.g., by functionals of type $(0)\sigma$, with lh, $*$, $[\ ]$ and $^{-}$ primitive recursive. Now we are able to formulate Howard's schema of bar recursion:

$$(BR^H)\qquad \underset{\sim}{B}^H\alpha=\begin{cases}G\alpha & \text{if } Y[\alpha]<lh\alpha,\\ H(\lambda X\cdot\underset{\sim}{B}^H(\alpha*X))\alpha & \text{else}.\end{cases}$$

2.2. The equivalence of B^S and B^H is obtained as follows.

Define $B\equiv\lambda YGHCx\cdot B^H YG^H H^H(\bar{C}x)$, with $G^H\equiv\lambda\alpha\cdot G[\alpha]lh\alpha$ and $H^H\equiv\lambda Z\alpha\cdot HZ[\alpha]lh\alpha$. Then by BR^H B satisfies:

$$\underset{\sim}{B}Cx=\begin{cases}G^H(\bar{C}x)=G[\bar{C}x]lh(\bar{C}x) & \text{if } Y[\bar{C}x]<lh(\bar{C}x),\\ H^H(\lambda X\cdot\underset{\sim}{B}^H(\bar{C}x*X))(\bar{C}x) & \text{else}.\end{cases}$$

Since $lh(\bar{C}x)=x$, $[\bar{C}x]=[C]_x$ (provable by induction) and $\overline{[C*_xX]_{x+1}}(x+1)=\bar{C}x*X$, it follows by extensionality that $Y[\bar{C}x]=Y[C]_x$ and $\underset{\sim}{B}[C*_xX]_{x+1}(x+1)=\underset{\sim}{B}^H(\bar{C}x*X)$. Hence by the definition of H^H and again by extensionality it follows that

$$H^H(\lambda X\cdot\underset{\sim}{B}^H(\bar{C}x*X))(\bar{C}x)=H(\lambda X\cdot\underset{\sim}{B}[C*_xX]_{x+1}(x+1))[C]_xx.$$

Hence by extensionality B satisfies BR^S:

$$\underset{\sim}{B}Cx=\begin{cases}G[C]_xx & \text{if } Y[C]_x<x,\\ H(\lambda X\cdot\underset{\sim}{B}[C*_xX]_{x+1}(x+1))[C]_xx & \text{else}.\end{cases}$$

For the converse define $B\equiv\lambda YGH\alpha\cdot B^S YG^S H^S[\alpha]lh\alpha$, with $G^S\equiv\lambda Cx\cdot G(\bar{C}x)$ and $H^S\equiv\lambda ZCx\cdot HZ(\bar{C}x)$. Then by BR^S B satisfies:

$$\underset{\sim}{B}\alpha=\begin{cases}G^S[[\alpha]]_{lh\alpha}lh\alpha & \text{if } Y[[\alpha]]_{lh\alpha}<lh\alpha,\\ H^S(\lambda X\cdot\underset{\sim}{B}^S[[\alpha]*_{lh\alpha}X]_{1+lh\alpha}(1+lh\alpha))[[\alpha]]_{lh\alpha}lh\alpha & \text{else}.\end{cases}$$

Since $[[\alpha]]_{lh\alpha}=[\alpha]$ and $[\alpha*X]=[[\alpha]*_{lh\alpha}X]_{1+lh\alpha}$ (both provable by induction), it follows by extensionality that $Y[[\alpha]]_{lh\alpha}=Y[\alpha]$ and

$$\underset{\sim}{B}(\alpha*X)=\underset{\sim}{B}^S[\alpha*X]lh(\alpha*X)=\underset{\sim}{B}^S[[\alpha]*_{lh\alpha}X]_{1+lh\alpha}(1+lh\alpha).$$

Moreover $\overline{[\alpha]}lh\alpha=\alpha$, hence by the definition of G^S and H^S and again by extensionality it follows that B satisfies BR^H:

$$\underset{\sim}{B}\alpha=\begin{cases}G\alpha & \text{if } Y[\alpha]<lh\alpha,\\ H(\lambda X\cdot\underset{\sim}{B}(\alpha*X))\alpha & \text{else}.\end{cases}$$

2.3. In [V] and [B] the representation of finite sequences differs from [S]. It turns out to be useful to show first that B^S and B_1, the bar recursor obtained from B^S by using another representation of finite sequences, are equivalent.

Corresponding to $[\alpha]$ and $[C]_x$ we define primitive recursively $\{\alpha\}$ and $\{C\}$ such that

$$y<lh\alpha\to\{\alpha\}y=\alpha_y,\qquad y\geqq lh\alpha>0\to\{\alpha\}y=\alpha_{lh\alpha-1},\qquad \{\langle\,\rangle\}y=0^\sigma,$$

and

$$y<x\to\{C\}_xy=Cy,\qquad y\geqq x>0\to\{C\}_xy=C(x-1),\qquad \{C\}_0y=0^\sigma.$$

Let B_1 abd B_1^H be constants with defining equations BR_1 resp. BR_1^H obtained by replacing $[\,]$ by $\{\,\}$ in BR^S resp. BR^H. The equivalence of B_1 and B_1^H follows immediately from 2.2 by replacing everywhere $[\,]$ by $\{\,\}$. B^H and B_1^H are interchangeable with respect to [H]: Howard only requires $[\alpha]$ to be a functional extending α "in some systematic way (by primitive recursion)".

We prefer to prove the equivalence of B^S and B_1 instead of B^H and B_1^H. For the construction of B^S from B_1 we can use essentially the same argument as in [B]. As pointed out by Howard, the converse can be obtained by an adaptation of this argument.

For $C\in(0)\sigma$ we define primitive recursively C^+ and C^- by

$$C^+\equiv\lambda x\cdot(C0+\ldots+Cx)\quad\text{and}\quad C^-0=C0,$$
$$C^-x=Cx\dot{-}C(x-1)\quad\text{for}\quad x>0.$$

When $\sigma=0$ these definitions are unproblematic, when $\sigma\neq 0$ the operators $\dot{-}$ (cut off subtraction) and $+$ are hereditarily defined: $T_1+T_2\equiv\lambda X\cdot(T_1X+T_2X)$ $T_1\dot{-}T_2\equiv\lambda X\cdot(T_1X\dot{-}T_2X)$. By induction we have $C^{+-}=C$ and $\{C^+\}_x^-=[C]_x$.

Define $B\equiv\lambda YGHCx\cdot B_1Y_1G_1H_1(C^+)x$, with $Y_1\equiv\lambda C\cdot Y(C^-)$, $G_1\equiv\lambda Cx\cdot G(C^-)x$ and $H_1\equiv\lambda ZCx\cdot H(\lambda X\cdot Z(Cx+X))(C^-)x$. Then by BR_1 B satisfies:

$$\underset{\sim}{B}Cx=\begin{cases}G_1\{C^+\}_xx=G(\{C^+\}_x^-)x & \text{if}\quad Y_1\{C^+\}_x<x\\ H_1(\lambda X\cdot\underset{\sim}{B}_1\{C^+*_xX\}_{x+1}(x+1))\{C^+\}_xx & \\ \quad=H(\lambda X\cdot\underset{\sim}{B}_1\{C^+*_x(\{C^+\}_xx+X)\}_{x+1}(x+1))(\{C^+\}_x^-)x & \\ & \text{otherwise}.\end{cases}$$

Since $\{C^+\}_x^-=[C]_x$ it follows by extensionality that $Y_1\{C^+\}_x=Y(\{C^+\}_x^-)=Y[C]_x$ and $G_1\{C^+\}_xx=G[C]_xx$. Since $\{C^+*_x(\{C^+\}_xx+X)\}_{x+1}=([C*_xX]_{x+1})^+$ (provable by induction) and hence by extensionality

$$\underset{\sim}{B}[C*_xX]_{x+1}(x+1)=\underset{\sim}{B}_1(([C*_xX]_{x+1})^+)(x+1)$$
$$=\underset{\sim}{B}_1\{C^+*_x(\{C^+\}_xx+X)\}_{x+1}\,(x+1),$$

it follows again by extensionality that B satisfies BR^S:

$$\underset{\sim}{B}Cx=\begin{cases}G[C]_xx & \text{if}\quad Y[C]_x<x,\\ H(\lambda X\cdot\underset{\sim}{B}[C*_xX]_{x+1}(x+1))[C]_xx & \text{else}.\end{cases}$$

For the converse we would like to apply an argument similar to the one above. A problem is that we are not allowed to use negative numbers and, as a consequence, $\dot{-}$ must be cut-off subtraction. But if we could replace $\dot{-}$ by ordinary subtraction we would have $C^{-+}=C$ and $[C^-]_x^+=\{C\}_x$. So the problem is overcome if we encode integers as natural numbers, e.g., by interpreting $2x$ as x and $2x+1$ as $-(x+1)$. Then operations $+$ and $-$ for integers can easily be defined by primitive recursion. Let also C^+ and C^- be redefined for integers.

Define $B \equiv \lambda YGHCx \cdot B^S Y^S G^S H^S(C^-)x$, with $Y^S \equiv \lambda C \cdot Y(C^+)$, $G^S \equiv \lambda Cx \cdot G(C^+)x$ and $H^S \equiv \lambda ZCx \cdot H(\lambda X \cdot Z(X-(C^+)x))(C^+)x$. Since $[C^-]_x^+ = \{C\}_x$ (provable by induction), we have by extensionality that

$$Y^S[C^-]_x = Y([C^-]_x^+) = Y\{C\}_x, \qquad G^S[C^-]_x x = G([C^-]_x^+)x = G\{C\}_x x$$

and

$$\begin{aligned} &H^S(\lambda X \cdot \underset{\sim}{B}^S[C^- *_x X]_{x+1}(x+1))[C^-]_x x \\ &\quad = H(\lambda X \cdot \underset{\sim}{B}^S[C^- *_x (X-([C^-]_x^+)x)]_{x+1}(x+1))([C^-]_x)x \\ &\quad = H(\lambda X \cdot \underset{\sim}{B}^S[C^- *_x (X-\{C\}_x x)]_{x+1}(x+1))\{C\}_x x . \end{aligned}$$

Since

$$\underset{\sim}{B}\{C *_x X\}_{x+1}(x+1) = \underset{\sim}{B}^S((\{C *_x X\}_{x+1})^-)(x+1)$$

and

$$(\{C *_x X\}_{x+1})^- = [C^- *_x (X-\{C\}_x x)]_{x+1},$$

it follows by BR^S and again by extensionality that B satisfies BR_1.

§ 3. Spector↔Tait

For his computational analysis, Tait considers B^T only as a combinator, but the corresponding schema BR^T of defining equations is easily read off from the conversion rules:

$$(BR^T) \qquad \underset{\sim}{B}^T Cx = \begin{cases} G & \text{if } \; Y[C]_x < x, \\ H(\lambda X \cdot \underset{\sim}{B}^T[C *_x X]_{x+1}(x+1)) & \text{else}. \end{cases}$$

B^T is different from B^S since G and H appear with fewer arguments. Equivalence is obtained as follows.

Define $B \equiv \lambda YGHCx \cdot B^S Y G^S H^S Cx$, with $G^S \equiv \lambda Cx \cdot G$ and $H^S \equiv \lambda ZCx \cdot HZ$. Then B trivially satisfies BR^T by BR^S (just throwing away arguments).

For the converse, define $B \equiv \lambda YGHCx \cdot B^T YGH^T Cx[C]_x x$, with $H^T \equiv \lambda ZCx \cdot H(\lambda X \cdot ZX[C *_x X]_{x+1}(x+1))Cx$. Then by BR^T B satisfies:

$$\underset{\sim}{B}Cx = \begin{cases} G[C]_x x & \text{if } \; Y[C]_x < x \\ H^T(\lambda X \cdot \underset{\sim}{B}^T[C *_x X]_{x+1}(x+1))[C]_x x & \\ \quad = H(\lambda X \cdot \underset{\sim}{B}^T[C *_x X]_{x+1}(x+1)[[C]_x *_x X]_{x+1}(x+1))[C]_x x & \\ & \text{else}. \end{cases}$$

Since $[[C *_x X]_{x+1} = [[C]_x *_x X]_{x+1}$ (provable by induction), it follows by extensionality that

$$\begin{aligned} \underset{\sim}{B}[C *_x X]_{x+1}(x+1) &= \underset{\sim}{B}^T[C *_x X]_{x+1}(x+1)[[C *_x X]_{x+1}]_{x+1}(x+1) \\ &= \underset{\sim}{B}^T[C *_x X]_{x+1}(x+1)[[C]_x *_x X]_{x+1}(x+1). \end{aligned}$$

Hence again by extensionality it follows that B satisfies BR^S:

$$\underset{\sim}{B}Cx = \begin{cases} G[C]_x x & \text{if } \; Y[C]_x < x, \\ H(\lambda X \cdot \underset{\sim}{B}[C *_x X]_{x+1}(x+1))[C]_x x & \text{else}. \end{cases}$$

§ 4. Spector↔Vogel

The equivalence of B^S and B_1 has already been established in Sect. 2. We prefer to compare B^V with B_1, since Vogel uses the same representation of finite sequences as used in BR_1. The schema BR^V corresponding to Vogel's conversion rules is:

$$\underset{\sim}{B}^V CxU = \begin{cases} G\{C *_x U\}_{x+1}(x+1) & \text{if} \quad Y\{C *_x U\}_{x+1} \leqq x, \\ H(\underset{\sim}{B}^V\{C *_x U\}_{x+1}(x+1))\{C *_x U\}_{x+1}(x+1) & \text{else}. \end{cases}$$

Equivalence of B^V and B_1 is obtained as follows.

Define $B \equiv \lambda YGHCxU \cdot B_1 YGH\{C *_x U\}_{x+1}(x+1)$, then by BR_1 B satisfies:

$$\underset{\sim}{B}CxU = \begin{cases} G\{\{C *_x U\}_{x+1}\}_{x+1}(x+1) \\ \qquad\qquad\qquad \text{if} \quad Y\{\{C *_x U\}_{x+1}\}_{x+1} < x+1, \\ H(\lambda X \cdot \underset{\sim}{B}_1\{\{C *_x U\}_{x+1} *_{x+1} X\}_{x+2}(x+2))\{\{C *_x U\}_{x+1}\}_{x+1}(x+1) \\ \qquad\qquad\qquad \text{otherwise}. \end{cases}$$

By induction we have $\{\{C *_x U\}_{x+1}\}_{x+1} = \{C *_x U\}_{x+1}$. By extensionality we have

$$\begin{aligned} \underset{\sim}{B}\{C *_x U\}_{x+1}(x+1) &= \lambda X \cdot \underset{\sim}{B}\{C *_x U\}_{x+1}(x+1)X \\ &= \lambda X \cdot \underset{\sim}{B}_1\{\{C *_x U\}_{x+1} *_{x+1} X\}_{x+2}(x+2). \end{aligned}$$

It follows again by extensionality that B satisfies BR^V:

$$\underset{\sim}{B}CxU = \begin{cases} G\{C *_x U\}_{x+1}(x+1) & \text{if} \quad Y\{C *_x U\}_{x+1} \leqq x, \\ H(\underset{\sim}{B}\{C *_x U\}_{x+1}(x+1))\{C *_x U\}_{x+1}(x+1) & \text{else}. \end{cases}$$

For the converse, define $B \equiv \lambda YGHCx \cdot B^V YGH\{C\}_{x \dot{-} 1}(x \dot{-} 1)(C(x \dot{-} 1))$. Then by BR^V B satisfies for all $x > 0$:

$$\underset{\sim}{B}Cx = \begin{cases} G\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x x & \text{if} \quad Y\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x \leqq x-1, \\ H(\underset{\sim}{B}^V\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x x)\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x x \\ \qquad\qquad\qquad \text{else}. \end{cases}$$

Since $\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x = \{\{C *_x X\}_{x+1}\}_x$ (provable by induction), we have by extensionality

$$\begin{aligned} \underset{\sim}{B}^V\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x x &= \lambda X \cdot \underset{\sim}{B}^V\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x xX \\ &= \lambda X \cdot \underset{\sim}{B}^V\{\{C *_x X\}_{x+1}\}_x xX = \lambda X \cdot \underset{\sim}{B}\{C *_x X\}_{x+1}(x+1). \end{aligned}$$

Moreover, by induction we have $\{\{C\}_{x-1} *_{x-1} C(x-1)\}_x = \{C\}_x$. It follows again by extensionality that B satisfies BR_1 for all $x > 0$:

$$\underset{\sim}{B}Cx = \begin{cases} G\{C\}_x x & \text{if} \quad Y\{C\}_x < x, \\ H(\lambda X \cdot \underset{\sim}{B}\{C *_x X\}_{x+1}(x+1))\{C\}_x x & \text{otherwise}. \end{cases}$$

So if we redefine B for $x = 0$ by

$$BYGHC0 = H(\lambda X \cdot BYGH\{C *_0 X\}_1 1)\{C\}_0 0$$

it follows that B satisfies BR_1 for all x.

§ 5. Spector↔Bezem

In [B] the following schema BR^B is used:

$$(BR^B)\quad \underset{\sim}{B}^B Cx = \begin{cases} GCx & \text{if } Y\{C\}_{x+1} < x, \\ H(\lambda X \cdot \underset{\sim}{B}^B\{C *_{x+1} X\}_{x+2}(x+1))Cx & \text{else}. \end{cases}$$

Equivalence of B^B and B_1 is obtained as follows. Let $0 * C$, for C of type $(0)\sigma$, be defined by:

$$x=0 \to (0 * C)x = 0^\sigma, \qquad x>0 \to (0 * C)x = C(x-1).$$

Define $B \equiv \lambda YGHCx \cdot B^B Y^B G^B H^B \{0 * C\}_{x+1} x$, with $Y^B \equiv \lambda C \cdot Y(\lambda y \cdot C(y+1))$, $G^B \equiv \lambda Cx \cdot G(\lambda y \cdot C(y+1))x$ and $H^B \equiv \lambda ZCx \cdot HZ\{\lambda y \cdot C(y+1)\}_x x$. Then by BR^B B satisfies:

$$\underset{\sim}{B}Cx = \begin{cases} G^B\{0 * C\}_{x+1}x & \text{if } Y^B\{\{0 * C\}_{x+1}\}_{x+1} < x, \\ H^B(\lambda X \cdot \underset{\sim}{B}^B\{\{0 * C\}_{x+1} *_{x+1} X\}_{x+2}(x+1))\{0 * C\}_{x+1}x & \\ & \text{else}. \end{cases}$$

By induction we can prove $\{\{0 * C\}_{x+1}\}_{x+1} = \{0 * C\}_{x+1}$ and $\{C\}_x = \lambda y \cdot \{0 * C\}_{x+1}(y+1)$. Hence it follows by extensionality that

$$Y^B\{\{0 * C\}_{x+1}\}_{x+1} = Y(\lambda y \cdot \{0 * C\}_{x+1}(y+1)) = Y\{C\}_x$$

and

$$G^B\{0 * C\}_{x+1}x = G(\lambda y \cdot \{0 * C\}_{x+1}(y+1))x = G\{C\}_x x.$$

Since also

$$\{0 * \{C *_x X\}_{x+1}\}_{x+2} = \{\{0 * C\}_{x+1} *_{x+1} X\}_{x+2}$$

(provable by induction), we have by extensionality

$$\begin{aligned} \underset{\sim}{B}\{C *_x X\}_{x+1}(x+1) &= \underset{\sim}{B}^B\{0 * \{C *_x X\}_{x+1}\}_{x+2}(x+1) \\ &= \underset{\sim}{B}^B\{\{0 * C\}_{x+1} *_{x+1} X\}_{x+2}(x+1). \end{aligned}$$

It follows again by extensionality that B satisfies BR_1:

$$\underset{\sim}{B}Cx = \begin{cases} G\{C\}_x x & \text{if } Y\{C\}_x < x, \\ H(\lambda X \cdot \underset{\sim}{B}\{C *_x X\}_{x+1}(x+1))\{C\}_x x & \text{else}. \end{cases}$$

For the converse, define $B \equiv \lambda YGHCx \cdot B_1 Y_1 G_1 H_1 C(x+1)C$, with $Y_1 \equiv \lambda C \cdot YC+1$, $G_1 \equiv \lambda DyC \cdot GC(y \dot- 1)$ and $H_1 \equiv \lambda ZDyC \cdot H(\lambda X \cdot ZX\{C *_y X\}_{y+1})C(y \dot- 1)$.

Then by BR_1 B satisfies:

$$\underset{\sim}{B}Cx = \begin{cases} G_1\{C\}_{x+1}(x+1)C = GCx & \text{if } Y_1\{C\}_{x+1} < x+1, \\ H_1(\lambda X \cdot \underset{\sim}{B}_1\{C *_{x+1} X\}_{x+2}(x+2))\{C\}_{x+1}(x+1)C & \\ \quad = H(\lambda X \cdot \underset{\sim}{B}_1\{C *_{x+1} X\}_{x+2}(x+2)\{C *_{x+1} X\}_{x+2})Cx & \\ & \text{otherwise}. \end{cases}$$

By the definition of B we have

$$\underset{\sim}{B}\{C *_{x+1} X\}_{x+2}(x+1) = \underset{\sim}{B}_1\{C *_{x+1} X\}_{x+2}(x+2)\,\{C *_{x+1} X\}_{x+2}\,.$$

Since $Y_1 \equiv \lambda C \cdot YC+1$ it follows by extensionality that B satisfies BR^B:

$$\underset{\sim}{B}Cx = \begin{cases} GCx & \text{if} \quad Y\{C\}_{x+1} < x\,, \\ H(\lambda X \cdot \underset{\sim}{B}\{C *_{x+1} X\}_{x+2}(x+1))Cx & \text{else}\,. \end{cases}$$

References

[Ba] Barendregt, H.P.: The lambda calculus. Amsterdam: North-Holland 1984

[B] Bezem, M.A.: Strongly majorizable functionals of finite type: a model for bar recursion containing discontinuous functionals. J. Symb. Logic **50**, 652–660 (1985)

[CF] Curry, H.B., Feys, R.: Combinatory logic. Amsterdam: North-Holland 1958

[H] Howard, W.A.: Functional interpretation of bar induction by bar recursion. Compos. Math. **20**, 107–124 (1968)

[L] Luckhardt, H.: Extensional Gödel functional interpretation. (Lect. Notes Math., vol. 306) Berlin Heidelberg New York: Springer 1973

[S] Spector, C.: Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In: Dekker, J.C.E. (ed.): Proceedings of symposia in pure mathematics V, pp. 1–27. Providence: AMS 1962

[TA] Tait, W.W.: Normal form theorem for bar recursive functions of finite type. In: Proceedings of the second scandinavian logic symposium, pp. 353–367. Amsterdam: North-Holland 1971

[T] Troelstra, A.S. (ed.): Metamathematical investigation of intuitionistic arithmetic and analysis. (Lect. Notes Math., vol. 344) Berlin Heidelberg New York: Springer 1973

[V] Vogel, H.: Ein starker Normalisationssatz für die barrekursiven Funktionale. Arch. Math. Logik Grundlagenforsch. **18**, 81–84 (1976)

Received August 31, 1987