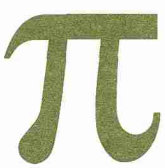
Department of Philosophy - Utrecht University

Bar Recursion versus Polymorphism

Extended abstract



Erik Barendsen
Marc Bezem

Logic Group
Preprint Series
No. 69
December 1991

Department of Philosophy
University of Utrecht
Heidelberglaan 8
3584 CS Utrecht
The Netherlands

Bar Recursion versus Polymorphism

Extended abstract

Erik Barendsen
University of Nijmegen*

Marc Bezem
University of Utrecht†

1. Introduction

By constructing a *counter model* we show that a certain appealing equation E has no solution in Girard's [1972] second order lambda calculus (= **λ2T**, or the *polymorphic* lambda calculus). The counter model is a new PER model based on an applicative structure consisting of untyped lambda calculus extended with a specific higher order oracle in the style of Kleene [1962]. A novelty of this model is the presence of discontinuous functionals, which seem to be absent in all known models of **λ2T**. The model construction is interesting in itself and can be applied more widely to obtain independence and consistency results.

The equation E = E(Φ) (with Φ a type **3** variable) is a simple functional equation in the language of Gödel's [1958] system of higher order primitive recursive functionals (= **λT**). The computational interpretation of E(Φ) is that Φ performs a certain minimization.

In contrast to the negative result concerning **λ2T** we show that E has a solution in Spector's [1962] system of *bar recursive* functionals (= **λTB**, the extension of **λT** with bar recursion). This indicates that **λTB** is in some sense stronger than **λ2T**.

Surprisingly, E has a solution in **λT**, but only provably so in **λTB** (thus showing that **λTB** is not conservative over **λT**).

We stress the fact that that our results show that **λ2T** and **λTB** have different classes of definable *functionals* (at least of type **3**), whereas the meta-mathematical results from Spector [1962] and Girard [1972] show that they have the same class of definable *functions*, namely the provably total functions of analysis.

*Department of Computer Science, Postbus 9010, 6500 GL Nijmegen, The Netherlands, e-mail `erikb@cs.kun.nl`, fax +31.80.553450.

†Department of Philosophy, Postbus 80126, 3508 TC Utrecht, The Netherlands, e-mail `bezem@phil.ruu.nl`, fax +31.30.532816.

2. Syntax

2.1. INTRODUCTION. We present the syntax of the following equational systems based on simply typed lambda calculus: $\boldsymbol{\lambda}^\tau, \boldsymbol{\lambda T}, \boldsymbol{\lambda TB}$ and $\boldsymbol{\lambda 2}, \boldsymbol{\lambda 2T}$. To the systems $\boldsymbol{\lambda T}, \boldsymbol{\lambda TB}, \boldsymbol{\lambda 2T}$ we can add quantifier-free arithmetic, which will be expressed by the denotation $\boldsymbol{\lambda\Box_A}$. With $\longrightarrow$ expressing the subsystem relation we can depict the situation as follows.

$$\begin{array}{ccccc} \boldsymbol{\lambda}^\tau & \longrightarrow & \boldsymbol{\lambda T}_{(\mathbf{A})} & \longrightarrow & \boldsymbol{\lambda TB}_{(\mathbf{A})} \quad \text{first order systems} \\ \downarrow & & \downarrow & & \\ \boldsymbol{\lambda 2} & \longrightarrow & \boldsymbol{\lambda 2T}_{(\mathbf{A})} & & \quad \text{second order systems} \end{array}$$

2.2. TYPES. The first and second order systems are distinguished by their respective sets of *types*, $\mathbb{T}_1, \mathbb{T}_2$, given by the following abstract syntax.

$$\begin{aligned} \mathbb{T}_1 &= \mathbf{0} \mid \mathbb{T}_1 \to \mathbb{T}_1, \\ \mathbb{T}_2 &= \mathbf{0} \mid \mathbb{V} \mid \mathbb{T}_2 \to \mathbb{T}_2 \mid \forall \mathbb{V}.\mathbb{T}_2. \end{aligned}$$

Here $\mathbb{V} = \{\alpha, \beta, \alpha', \ldots\}$ is an infinite set of *type variables*. Note that $\mathbb{T}_1 \subset \mathbb{T}_2$. In the sequel, $\sigma, \tau, \sigma', \ldots$ range over $\mathbb{T}_2$. It is convenient to single out some special $\mathbb{T}_1$ types: $\mathbf{1} \equiv \mathbf{0}\to\mathbf{0}$, $\mathbf{2} \equiv \mathbf{1}\to\mathbf{0}$, and so on.

2.3. TERMS. (i) The systems $\boldsymbol{\lambda}^\tau$ and $\boldsymbol{\lambda 2}$ are the well-known simply typed lambda calculus and its second order (polymorphic) extension. Typical examples of $\boldsymbol{\lambda}^\tau$-terms are: $x^{\mathbf{0}}$, $f^{\mathbf{1}}$, $(\lambda f^{\mathbf{1}} x^{\mathbf{0}}.f(fx))(\lambda y^{\mathbf{0}}.y)$. Typical examples of $\boldsymbol{\lambda 2}$-terms are: all typical examples of $\boldsymbol{\lambda}^\tau$-terms, $\Lambda\alpha\lambda x^\alpha.x$ (the so-called polymorphic identity), $(\Lambda\alpha\lambda f^{\alpha\to\alpha} x^\alpha.f(fx))(\forall\alpha.\sigma)$.

(ii) The systems $\boldsymbol{\lambda T}_{(\mathbf{A})}$, $\boldsymbol{\lambda TB}_{(\mathbf{A})}$ (resp. $\boldsymbol{\lambda 2T}_{(\mathbf{A})}$) are based on $\boldsymbol{\lambda}^\tau$ (resp. $\boldsymbol{\lambda 2}$), but allow the occurrence of certain typed *constants* in the terms. These constants are

$$\begin{aligned} \boldsymbol{0}, \boldsymbol{S}, \boldsymbol{R}_\sigma \quad &\text{for the systems} \quad \boldsymbol{\lambda T}_{(\mathbf{A})}, \boldsymbol{\lambda TB}_{(\mathbf{A})}, \boldsymbol{\lambda 2T}_{(\mathbf{A})}, \\ \boldsymbol{B} \quad &\text{for the systems} \quad \boldsymbol{\lambda TB}_{(\mathbf{A})}. \end{aligned}$$

Here $\boldsymbol{0}$ is of type $\mathbf{0}$ and $\boldsymbol{S}$ is of type $\mathbf{1}$, with intended interpretation *zero* respectively the *successor* function (the intended interpretation of $\mathbf{0}$ is the set of natural numbers). We use the abbreviations $\boldsymbol{1} \equiv \boldsymbol{S0}$, $\boldsymbol{2} \equiv \boldsymbol{S}(\boldsymbol{S0})$, and so on. The constants $\boldsymbol{R}_\sigma$ of type $\sigma\to(\mathbf{0}\to\sigma\to\sigma)\to\mathbf{0}\to\sigma$ are added for all types σ of the system in question; their intended interpretation is that of *primitive recursor*. The type of the constant $\boldsymbol{B}$ is $\mathbf{2}\to\mathbf{1}\to(\mathbf{1}\to\mathbf{0}\to\mathbf{0})\to\mathbf{0}\to\mathbf{0}$; intended interpretation of $\boldsymbol{B}$ is *bar recursor*. A typical example of a $\boldsymbol{\lambda T}$-term is $\mathbf{Cond} \equiv \lambda x^{\mathbf{0}} y^{\mathbf{0}} z^{\mathbf{0}}.\boldsymbol{R_0} z(\lambda n^{\mathbf{0}} m^{\mathbf{0}}.y)x$. In the sequel, $\mathrm{Term}_\sigma(\boldsymbol{\lambda\Box})$ stands for the set of $\boldsymbol{\lambda\Box}$-terms of type σ. $F, M, N, M', \ldots$ range over terms. We shall tacitly assume that terms are well-typed and shall reduce type superscripts to a minimum that is needed to reconstruct the types of the constituents of a well-typed term.

2.4. EQUATIONS. The formulas of $\boldsymbol{\lambda}\Box$ are *equations* $M =_\sigma N$ with σ a type of $\boldsymbol{\lambda}\Box$ and M and N terms of $\boldsymbol{\lambda}\Box$ of type σ. Typical examples of such equations can be found in the inference rules that define the theories $\boldsymbol{\lambda}\Box$ below. In the sequel, $E, E', E(x,y), \ldots$ range over equations. We shall omit the type subscript in equations when confusion is not likely.

2.5. THEORY. The theories $\boldsymbol{\lambda}\Box$ are built up from axioms and rules that naturally divide into three groups.

1. *Lambda calculus* axioms and rules. We distinguish the following subgroups.

   (L1) Basic axioms and rules for simply typed lambda calculus (first order).

   (β_1) $(\lambda x^\sigma.M)N =_\tau M[x := N]$,

   (η_1) $\lambda x^\sigma.Mx =_{\sigma\to\tau} M$ provided $x \notin \mathrm{FV}(M)$.

   Equality and compatibility axioms/rules (cf. Barendregt [1984]).

   (L2) Axioms and rules for polymorphism (second order).

   (β_2) $(\Lambda\alpha.M)\tau =_{\sigma[\alpha:=\tau]} M[\alpha := \tau]$,

   (η_2) $\Lambda\alpha.M\alpha =_{\forall\alpha.\sigma} M$ (optional, not used in this paper).

   Equality and compatibility axioms/rules extended to the second order case.

2. Defining equations for *constants*, divided into the following subgroups.

   (CR) $\boldsymbol{R}_\sigma MN\boldsymbol{0} =_\sigma M$, $\quad \boldsymbol{R}_\sigma MN(\boldsymbol{S}P) =_\sigma NP(\boldsymbol{R}_\sigma MNP)$.

   (CB) Rules for the bar recursor, to be explained below.

$$\frac{Y[s] < \mathrm{lh}(s)}{\boldsymbol{B}YGHs = Gs} \qquad \frac{Y[s] \geq \mathrm{lh}(s)}{\boldsymbol{B}YGHs = H(\lambda x^{\boldsymbol{0}}.\boldsymbol{B}YGH(s * \langle x\rangle))s}$$

3. Additional rules for *quantifier-free arithmetic* (induction rule will be explained in the full paper).

(A) $$\frac{\boldsymbol{S}M = \boldsymbol{S}N}{M = N} \qquad \frac{\boldsymbol{S}M = \boldsymbol{0}}{E} \qquad \frac{E(\boldsymbol{0}, y) \qquad \begin{array}{c}\cancel{E(x, Fy)}\\ \vdots \\ E(\boldsymbol{S}x, y)\end{array}}{E(x,y)}$$

The (sub)groups L1, L2, CR, CB and A compose into the theories $\boldsymbol{\lambda}\Box$ according to the following table.

$\boldsymbol{\lambda}^\tau$	L1				
$\boldsymbol{\lambda}\mathbf{T}_{(\mathbf{A})}$	L1		CR		(A)
$\boldsymbol{\lambda}\mathbf{TB}_{(\mathbf{A})}$	L1		CR	CB	(A)
$\boldsymbol{\lambda 2}$	L1	L2			
$\boldsymbol{\lambda 2}\mathbf{T}_{(\mathbf{A})}$	L1	L2	CR		(A)

2.6. REMARKS. (i) We use the denotation $\langle n_0, \ldots, n_{k-1} \rangle$ $(k \geq 0)$ for finite sequences of natural numbers; the empty sequence is denoted by $\langle\,\rangle$. Let $s, s', \ldots$ range over such sequences. We presuppose a surjective, primitive recursive encoding of such sequences as natural numbers, with $*$ (concatenation operator), lh (length function) and $\preceq$ (prefix relation) primitive recursive. If $s = \langle n_0, \ldots, n_{k-1} \rangle$ then $[s]$ is the function assigning n_i to $i < k$ and 0 to $i \geq k$; $[s]$ is primitive recursive in s. The denotations $<, \geq, \mathrm{lh}(s), *, \langle x \rangle$, and $[s]$ in the defining rules of the bar recursor (CB above) should be taken as abbreviations of $\boldsymbol{\lambda}\mathbf{T}$-terms representing the corresponding primitive recursive functions and (characteristic functions of) relations explained above, e.g. $Y[s] < \mathrm{lh}(s)$ stands for $\boxed{<}(Y[s])(\mathrm{lh}(s)) = \boldsymbol{1}$. Thus the syntax in CB is explained.

(ii) Bar recursion (of type $\mathbf{0}$) is in fact recursion on trees of finite sequences (of natural numbers). As an example, consider for given Y, G, H the computation of $\boldsymbol{BYGH}\langle\,\rangle$. This involves recursion on the tree $T = \{\sigma \mid \forall \sigma' \prec \sigma\ Y[\sigma'] \geq \mathrm{lh}(\sigma')\}$. The termination condition of the recursion is $Y[s] < \mathrm{lh}(s)$: in a leaf s of T the value is given by Gs. In an inner node s of T the value of $\boldsymbol{BYGH}s$ is defined recursively in terms of all values $\boldsymbol{BYGH}(s * \langle x \rangle)$: by applying H to $\lambda x.\boldsymbol{BYGH}(s * \langle x \rangle)$. Of course Y should be such that T is well-founded, which indicates that it is non-trivial to find a model for bar recursion.

(iii) Let $\boldsymbol{\lambda}\Box$ be one of the systems $\boldsymbol{\lambda}\mathbf{T}, \boldsymbol{\lambda}\mathbf{TB}, \boldsymbol{\lambda}\mathbf{2T}$. The ease in use of the quantifier-free arithmetic of $\boldsymbol{\lambda}\Box_{\mathbf{A}}$ would be greatly improved by the addition of propositional logic. This can be achieved within the equational system since type $\mathbf{0}$ equality can be encoded: there exists a term $\boxed{=}$ of $\boldsymbol{\lambda}\mathbf{T}$ such that in $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$ both $M =_{\mathbf{0}} N \vdash \boxed{=}MN =_{\mathbf{0}} \boldsymbol{1}$ and $\boxed{=}MN =_{\mathbf{0}} \boldsymbol{1} \vdash M =_{\mathbf{0}} N$ (this cannot be done for $M =_\sigma N$ with $\sigma \neq \mathbf{0}$). Furthermore, the classical truth functions for propositional logic can be encoded in $\boldsymbol{\lambda}\mathbf{T}$ (with $\boldsymbol{1}$ representing 'true' and $\boldsymbol{0}$ representing 'false'). Thus propositional logic can be used (with type $\mathbf{0}$ equations), while the equational character of the theory is preserved. For example, $x > \boldsymbol{0} \to \mathbf{Cond}\, xyz = y$ stands for the equation

$$\boxed{\to}(\boxed{>}x\boldsymbol{0})(\boxed{=}(\mathbf{Cond}\, xyz)y) =_{\mathbf{0}} \boldsymbol{1},$$

where $\boxed{\to}$, $\boxed{>}$, $\boxed{=}$ are terms encoding $\to$, $>$ and $=_{\mathbf{0}}$ as explained previously. This equation is indeed provable in $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$. Moreover, bounded quantification $(\forall x \leq y\ \cdots, \exists x \leq y\ \cdots)$ can be represented in this way.

(iv) Using the encoding of formulas explained above, the term **Cond** can be used for case distinction (if $\cdots$ then $\cdots$ else $\cdots$). Also bounded minimization $(\mu x \leq y\ \cdots)$ can be implemented in $\boldsymbol{\lambda}\mathbf{T}$.

3. A new PER model

In this section a model construction for $\boldsymbol{\lambda}\mathbf{2T}_{\mathbf{A}}$ is presented. The construction will be used to obtain a certain counter model in section 4. The essence of this counter model is that it contains a specific discontinuous functional of type $\mathbf{3}$.

Well-known model constructions for $\boldsymbol{\lambda}\mathbf{2}$ and $\boldsymbol{\lambda}\mathbf{2T}$ seem to exhibit an inherent continuity. Even a relativized version of the recursion theoretic model HEO_2 (see Girard [1972]) does not work: one can show that the Kreisel-Lacombe-Schoenfield theorem can be relativized.

Our model is obtained using a PER-construction (see Breazu-Tannen and Coquand [1988]), based on an applicative structure consisting of untyped λ-terms, extended with a constant serving as a higher type oracle, modulo some conversion relation.

The idea of adding higher-type oracles to untyped λ-calculus originates from Kleene [1962]. He used the system to prove the equivalence between recursiveness and lambda definability in higher types. Below we describe a variant with one single oracle. The generalized construction, where an entire first order model is added in the form of oracles (to appear in a paper by the first author), however, appears to have interesting applications.

Untyped lambda calculus with higher type oracles

In this subsection the applicative structure will be constructed which forms the basis of a PER-model.

For this subsection, let $\varphi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ be a (type-**2**) functional.

3.1. DEFINITION. Let $\boldsymbol{\varphi}$ be a constant. The set of *$\lambda\varphi$-terms* (notation $\Lambda\varphi$) is defined inductively as follows.

$$\begin{aligned} \boldsymbol{\varphi} &\in \Lambda\varphi, \\ x \in V &\Rightarrow x \in \Lambda\varphi, \\ M, N \in \Lambda\varphi &\Rightarrow (MN) \in \Lambda\varphi, \\ M \in \Lambda\varphi, x \in V &\Rightarrow (\lambda x.M) \in \Lambda\varphi. \end{aligned}$$

We use the same notional conventions for $\lambda\varphi$-terms as we do for λ-terms; see Barendregt [1984]. The set of closed $\lambda\varphi$-terms is denoted as $\Lambda^{\circ}\varphi$.

The principle reduction relation is β-reduction. A second notion of reduction will be added to the system. We suppose the reader is familiar with the concept of reduction relations and induced conversion relations (denoted by $\to_\beta$ and $=_\beta$ for β-reduction).

3.2. DEFINITION. For each $n \in \mathbb{N}$ the *Church numeral* $\ulcorner n \urcorner \in \Lambda\varphi$ is defined by setting

$$\ulcorner n \urcorner \equiv \lambda f x.f^n(x).$$

3.3. DEFINITION. Let $\to_R$ be a notion of reduction (on $\Lambda\varphi$), and $F \in \Lambda\varphi, f : \mathbb{N} \to \mathbb{N}$. Then F is said to *R-define* f (notation $F \rhd_R f$) if for all $n \in \mathbb{N}$

$$F\ulcorner n \urcorner =_R \ulcorner f(n) \urcorner.$$

The aim is to construct a reduction relation $\to_\varphi$ such that for any F and f

$$\boldsymbol{\varphi} F \to_\varphi \ulcorner \varphi(f) \urcorner \qquad \text{if} \qquad F \rhd_{\beta\varphi} f. \tag{$*$}$$

Note that if '$F \rhd_{\beta\varphi} f$' would be replaced by '$F \rhd_\beta f$', the reduction $\to_\varphi$ could immediately be obtained by so-called δ-reduction making some external function internal (see Barendregt [1984]). This corresponds to the notion of

oracle in recursion theory. In $(*)$, however, the behaviour of the oracle is specified in terms of itself.

Because of this impredicativity, the question arises whether or not $\to_\varphi$ can be well defined. This is indeed the case. The idea is to generate $\to_\varphi$ in stages, according to an inductive operator. The details of this transfinite induction can be found in a forthcoming paper by the first author and J.P. van Draanen. The most important property of $\to_{\beta\varphi}$ (crucial in the proof of proposition 4.13) is the following (cf. Kleene [1962]).

3.4. THEOREM. *The reduction relation $\to_{\beta\varphi}$ is Church-Rosser.*

PROOF (Sketch). By a nested transfinite induction using the Hindley-Rosen lemma (see Barendregt [1984]). □

Building a PER model

We consider the set of closed $\lambda\varphi$-terms (modulo $=_{\beta\varphi}$) as an applicative structure.

3.5. DEFINITION. (i) For $M \in \Lambda^\circ\varphi$ write

$$\leqslant M \geqslant = \{M \in \Lambda^\circ\varphi \mid M =_{\beta\varphi} N\}.$$

(ii) $\mathcal{L}_\varphi = \{\leqslant M \geqslant \mid M \in \Lambda^\circ\varphi\}$.

(iii) *Application* in $\mathcal{L}_\varphi$ is defined as usual:

$$\leqslant M \geqslant \cdot \leqslant N \geqslant = \leqslant MN \geqslant.$$

Note that this is a sound definition, i.e. the resulting equivalence class is independent of the choice of the representatives of $\leqslant M \geqslant$ and $\leqslant N \geqslant$.

For the interpretation of $\boldsymbol{\lambda 2T}$-terms in a per structure over $\mathcal{L}_\varphi$ a few facts from classical untyped lambda calculus are needed.

To simplify notation, $=_{\beta\varphi}$ is mostly written as $=$.

3.6. DEFINITION. (i) Define

$$\begin{aligned} \mathbf{true} &\equiv \lambda xy.x, \\ \mathbf{false} &\equiv \lambda xy.y. \end{aligned}$$

Then $\mathbf{true}\,MN = M$ and $\mathbf{false}\,MN = N$, so if B is either equal to **true** or **false** the conditional if B then P else Q can be expressed as BPQ.

(ii) Let $\mathbf{S}^+, \mathbf{P}^-, \mathbf{Zero} \in \Lambda^\circ$ be a successor, a predecessor and a test for zero w.r.t. the numerals $\ulcorner n \urcorner$ (cf. Barendregt [1984]).

(iii) Using the fixed point combinator $\mathbf{Y}$, define

$$\mathbf{Rec} \equiv \mathbf{Y}(\lambda rmnx.\mathsf{if}\ \mathbf{Zero}\,x\ \mathsf{then}\ m\ \mathsf{else}\ n(\mathbf{P}^-x)(rmn(\mathbf{P}^-x))).$$

Now untyped terms with constants from $\boldsymbol{\lambda 2T}$ can be interpreted in $\mathcal{L}_\varphi$. This gives a way to interpret $\boldsymbol{\lambda 2T}$-terms.

3.7. INTERPRETATION. After replacing $\boldsymbol{0}$ by $\ulcorner 0 \urcorner$, $\boldsymbol{S}$ by $\mathbf{S}^+$, and $\boldsymbol{R}_\sigma$ by **Rec** untyped terms are interpreted as in ordinary term models. Then $\boldsymbol{\lambda 2T}$-terms are, as usual in PER-models, interpreted by erasing type information and using this untyped interpretation. The resulting model is called $\mathfrak{P}_\varphi$.

3.8. THEOREM. *$\mathfrak{P}_\varphi$ is a model of $\boldsymbol{\lambda 2T_A}$.*

4. Proof theoretic results

We will compare the first order systems $\boldsymbol{\lambda TB_A}$ and $\boldsymbol{\lambda T_A}$, and show that $\boldsymbol{\lambda TB_A}$ is a nonconservative extension of $\boldsymbol{\lambda T_A}$. This will be done by constructing an equation E in the language of $\boldsymbol{\lambda T_A}$ such that

$$\boldsymbol{\lambda TB_A} \vdash \mathrm{E},$$
$$\boldsymbol{\lambda T_A} \nvdash \mathrm{E}.$$

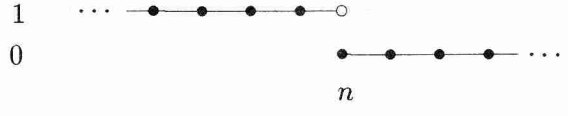
The nonconservativity does not really come as a surprise since it follows from Spector [1962] that $\boldsymbol{\lambda TB_A}$ proves the consistency of Peano Arithmetic, i.e. $\neg \mathrm{Proof}_{\mathbf{PA}}(x, \ulcorner 0 = 1 \urcorner)$ (after suitable encoding). The equation formulated here, however, does not refer to any metamathematical properties and is very simple.

The same equation yields a negative result concerning definability in $\boldsymbol{\lambda 2T_A}$.

4.1. DEFINITION. The term $\boldsymbol{\Sigma} \in \mathrm{Term}_{\mathbf{0}\to\mathbf{0}\to\mathbf{0}}(\boldsymbol{\lambda T})$ is defined by

$$\boldsymbol{\Sigma} \equiv \lambda xy.\mathsf{if}\ y < x\ \mathsf{then}\ \boldsymbol{1}\ \mathsf{else}\ \boldsymbol{0}.$$

The intuition is that $\boldsymbol{\Sigma n}$ is a so called *stepfunction* that can be depicted as follows.



4.2. DEFINITION. The equation $\mathrm{E} \equiv \mathrm{E}(\Phi)$, with Φ a type-**3** variable, is defined by

$$\mathrm{E}(\Phi) \equiv x < \Phi\varphi \to \varphi(\boldsymbol{\Sigma} x) \geq x \quad \wedge \quad \varphi(\boldsymbol{\Sigma}(\Phi\varphi)) < \Phi\varphi.$$

Here φ is a type-**2** variable and x a type-**0** variable. Note that $\mathrm{E}(\Phi)$ can be written in the form $G\Phi\varphi x =_{\mathbf{0}} \boldsymbol{0}$ or $G\Phi =_{\mathbf{2}\to\mathbf{0}\to\mathbf{0}} \lambda\varphi x.\boldsymbol{0}$.

4.3. INTUITION. If Φ satisfies $\mathrm{E}(\Phi)$ then

$$\Phi\varphi = \mu x\ [\varphi(\boldsymbol{\Sigma} x) < x].$$

4.4. DEFINITION. Let $\boldsymbol{\lambda\Box}$, $\boldsymbol{\lambda\boxtimes}$ be two of our systems. We say that E is $\boldsymbol{\lambda\Box}$-*realizable* in $\boldsymbol{\lambda\boxtimes}$ if for some $F \in \mathrm{Term}_{\mathbf{3}}(\boldsymbol{\lambda\Box})$ one has

$$\boldsymbol{\lambda\boxtimes} \vdash \mathrm{E}(F).$$

Our aim is to show that E is $\boldsymbol{\lambda}\mathbf{T}$-realizable in $\boldsymbol{\lambda}\mathbf{T}\mathbf{B}_\mathbf{A}$, but not in $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A}$.

4.5. PROPOSITION. *There exists no* $F \in \mathrm{Term}_3(\boldsymbol{\lambda}\mathbf{T})$ *such that*

$$\boldsymbol{\lambda}\mathbf{T}_\mathbf{A} \vdash \mathrm{E}(F).$$

PROOF. Suppose, towards a contradiction, $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A} \vdash \mathrm{E}(F)$ for some F. Now consider $\mathfrak{M}(\mathbb{N})$, the so-called full type structure of functionals over $\mathbb{N}$. Note that $\mathfrak{M}(\mathbb{N})$ is a model of $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A}$. Define $\delta : \mathbb{N} \to \mathbb{N}^\mathbb{N}$ by

$$\begin{aligned} \delta(n)(x) &= 1 \quad \text{if } x < n, \\ &= 0 \quad \text{if } x \geq n. \end{aligned}$$

Note that $[\![\boldsymbol{\Sigma}]\!] = \delta$. Notation: $\delta_n = \delta(n)$. Let $\varphi : \mathbb{N}^\mathbb{N} \to \mathbb{N}$ be such that for each $n \in \mathbb{N}$

$$\varphi(\delta_n) = n.$$

Then one has

$$\varphi(\delta([\![F]\!]\varphi)) = [\![F]\!]\varphi,$$

contradicting $\mathfrak{M}(\mathbb{N}) \models \mathrm{E}(F)$. $\square$

It can easily be seen that E is $\boldsymbol{\lambda}\mathbf{T}\mathbf{B}$-realizable in $\boldsymbol{\lambda}\mathbf{T}\mathbf{B}$. Indeed, the minimization stated above can be obtained by constructing a 'searching functional' Ψ of type $\mathbf{2}{\to}\mathbf{0}{\to}\mathbf{0}$ such that

$$\begin{aligned} \Psi\varphi s &= 0 && \text{if } \varphi[s] < \mathrm{lh}(s), \\ &= 1 + \Psi\varphi(s * \langle 1 \rangle) && \text{otherwise,} \end{aligned}$$

and finally defining $F \equiv \lambda\varphi.\Psi\varphi\langle\,\rangle$. From the form of the equations for Ψ it is clear that such a Ψ can be constructed using the bar recursor.

4.6. DEFINITION. The term $\mathrm{F_B} \in \mathrm{Term}_3(\boldsymbol{\lambda}\mathbf{T}\mathbf{B})$ is defined as follows.

$$\mathrm{F_B} \equiv \lambda\varphi.\boldsymbol{B}\varphi GH\langle\,\rangle,$$

where

$$\begin{aligned} G &\equiv \lambda s.\boldsymbol{0}, \\ H &\equiv \lambda fs.\boldsymbol{S}(f\boldsymbol{1}). \end{aligned}$$

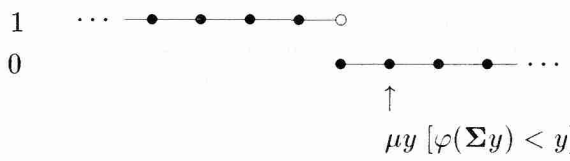
4.7. PROPOSITION. $\boldsymbol{\lambda}\mathbf{T}\mathbf{B}_\mathbf{A} \vdash \mathrm{E}(\mathrm{F_B})$.

It will turn out that E is even $\boldsymbol{\lambda}\mathbf{T}$-realizable in $\boldsymbol{\lambda}\mathbf{T}\mathbf{B}_\mathbf{A}$. Use is made of a trick due to G. Kreisel, also employed by J.A. Bergstra and J. Terlouw, see Barendregt [1984], p. 581.

4.8. DEFINITION. Define $f_\varphi \in \mathrm{Term}_1(\boldsymbol{\lambda}\mathbf{T})$ by

$$f_\varphi \equiv \lambda x.\mathsf{if}\ \forall y \leq x{+}1\ [\varphi(\boldsymbol{\Sigma}y) \geq y]\ \mathsf{then}\ \boldsymbol{1}\ \mathsf{else}\ \boldsymbol{0}.$$

4.9. INTUITION. f_φ is a stepfunction which looks as follows.



So $f_\varphi = \Sigma(\mu y\ [\varphi(\Sigma y) < y] - 1)$ (provided $\exists y\ \varphi(\Sigma y) < y$; otherwise '$f_\varphi = \Sigma_\infty$'). Therefore

$$\varphi(f_\varphi) \geq \mu y\ [\varphi(\Sigma y) < y] - 1$$

and hence

$$\mu y\ [\varphi(\Sigma y) < y] \leq \varphi(f_\varphi) + 1.$$

This gives a primitive recursive upperbound of the minimization expressed in E.

4.10. DEFINITION. The term $\mathrm{F_T} \in \mathrm{Term_3}(\boldsymbol{\lambda T})$ is defined as follows.

$$\mathrm{F_T} \equiv \lambda\varphi.\mu x \leq \varphi(f_\varphi) + 1\ [\varphi(\Sigma x) < x].$$

4.11. PROPOSITION. $\boldsymbol{\lambda TB_A} \vdash \mathrm{E}(\mathrm{F_T})$.

PROOF (Sketch). One shows that $\mathrm{F_T}\varphi = \mathrm{F_B}\varphi$, using the argument lined out in 4.9. Hence by extensionality one has $\mathrm{F_T} = \mathrm{F_B}$. Combining this with proposition 4.7 gives $\mathrm{E}(\mathrm{F_T})$. □

Now we can combine the results obtained thus far.

4.12. THEOREM. $\boldsymbol{\lambda TB_A}$ *is a non-conservative extension of* $\boldsymbol{\lambda T_A}$.

PROOF. Note that $\mathrm{E}(\mathrm{F_T})$ is an equation in the language of $\boldsymbol{\lambda T_A}$. By proposition 4.5

$$\boldsymbol{\lambda T_A} \nvdash \mathrm{E}(\mathrm{F_T}),$$

whereas by proposition 4.11

$$\boldsymbol{\lambda TB_A} \vdash \mathrm{E}(\mathrm{F_T}).\ \square$$

It will now be shown that E is not even $\boldsymbol{\lambda 2T}$-realizable in $\boldsymbol{\lambda 2T_A}$. This suggests that bar recursion is, in some sense, a more powerful extension of $\boldsymbol{\lambda T}$ than the concept of polymorphism.

The idea of the proof is the same as in the proof for $\boldsymbol{\lambda T_A}$ (proposition 4.5), but the counter model is far more involved. The presence of a type-**2** functional like φ, introducting a 'fatal discontinuity' would solve the problem. This can be achieved by applying the construction from section 3.

4.13. PROPOSITION. *There exists no* $F \in \mathrm{Term_3}(\boldsymbol{\lambda 2T})$ *such that*

$$\boldsymbol{\lambda 2T_A} \vdash \mathrm{E}(F).$$

PROOF (Sketch). Take $\varphi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ such that $\varphi(\delta_n) = n$ (like in the proof of proposition 4.5), and consider $\mathfrak{P}_\varphi$. The rest of the proof is analogous to the one for the first-order case, using the Church-Rosser theorem. □

References

BARENDREGT, H.P.

[1984] *The lambda calculus: its syntax and semantics*, 2nd edition, Studies in Logic 103, North-Holland, Amsterdam.

BREAZU-TANNEN, V. and TH. COQUAND

[1988] Extensional models for polymorphism, *Theor. Comput. Sci.* 59, pp. 85–114.

DEKKER, J.C.E. (ed.)

[1962] *Proceedings of symposia in pure mathematics V*, AMS, Providence.

GIRARD, J.-Y.

[1972] Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur, Thèse de doctorat d'état, Université Paris VII.

GÖDEL, K.

[1958] Über eine bisher noch nicht benutzte Erweiterung des finiten Standpunktes, *Dialectica* 12, pp. 280–287.

KLEENE, S.C.

[1962] Lambda-definable functionals of finite types, *Fundamenta Math.* 50, pp. 281–303.

SPECTOR, C.

[1962] Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics, in: DEKKER [1962], pp. 1–27.

Logic Group Preprint Series

Department of Philosophy, University of Utrecht
Heidelberglaan 8, 3584 CS Utrecht
The Netherlands

1 C.P.J. Koymans, J.L.M. Vrancken, *Extending Process Algebra with the empty process*, September 1985
2 J.A. Bergstra, *A process creation mechanism in Process Algebra*, September 1985
3 J.A. Bergstra, *Put and get, primitives for synchronous unreliable message passing*, October 1985
4 A. Visser, *Evaluation, provably deductive equivalence in Heyting's arithmetic of substitution instances of propositional formulas*, November 1985
5 G.R. Renardel de Lavalette, *Interpolation in a fragment of intuitionistic propositional logic*, January 1986
6 C.P.J. Koymans, J.C. Mulder, *A modular approach to protocol verification using Process Algebra*, April 1986
7 D. van Dalen, F.J. de Vries, *Intuitionistic free abelian groups*, April 1986
8 F. Voorbraak, *A simplification of the completeness proofs for Guaspari and Solovay's R*, May 1986
9 H.B.M. Jonkers, C.P.J. Koymans & G.R. Renardel de Lavalette, *A semantic framework for the COLD-family of languages*, May 1986
10 G.R. Renardel de Lavalette, *Strictheidsanalyse*, May 1986
11 A. Visser, *Kunnen wij elke machine verslaan? Beschouwingen rondom Lucas' argument*, July 1986
12 E.C.W. Krabbe, *Naess's dichotomy of tenability and relevance*, June 1986
13 H. van Ditmarsch, *Abstractie in wiskunde, expertsystemen en argumentatie*, Augustus 1986
14 A. Visser, *Peano's Smart Children, a provability logical study of systems with built-in consistency*, October 1986
15 G.R. Renardel de Lavalette, *Interpolation in natural fragments of intuitionistic propositional logic*, October 1986
16 J.A. Bergstra, *Module Algebra for relational specifications*, November 1986
17 F.P.J.M. Voorbraak, *Tensed Intuitionistic Logic*, January 1987
18 J.A. Bergstra, J. Tiuryn, *Process Algebra semantics for queues*, January 1987
19 F.J. de Vries, *A functional program for the fast Fourier transform*, March 1987
20 A. Visser, *A course in bimodal provability logic*, May 1987
21 F.P.J.M. Voorbraak, *The logic of actual obligation, an alternative approach to deontic logic*, May 1987
22 E.C.W. Krabbe, *Creative reasoning in formal discussion*, June 1987
23 F.J. de Vries, *A functional program for Gaussian elimination*, September 1987
24 G.R. Renardel de Lavalette, *Interpolation in fragments of intuitionistic propositional logic*, October 1987 (revised version of no. 15)
25 F.J. de Vries, *Applications of constructive logic to sheaf constructions in toposes*, October 1987
26 F.P.J.M. Voorbraak, *Redeneren met onzekerheid in expertsystemen*, November 1987
27 P.H. Rodenburg, D.J. Hoekzema, *Specification of the fast Fourier transform algorithm as a term rewriting system*, December 1987
28 D. van Dalen, *The war of the frogs and the mice, or the crisis of the Mathematische Annalen*, December 1987
29 A. Visser, *Preliminary Notes on Interpretability Logic*, January 1988
30 D.J. Hoekzema, P.H. Rodenburg, *Gauß elimination as a term rewriting system*, January 1988
31 C. Smoryński, *Hilbert's Programme*, January 1988
32 G.R. Renardel de Lavalette, *Modularisation, Parameterisation, Interpolation*, January 1988
33 G.R. Renardel de Lavalette, *Strictness analysis for POLYREC, a language with polymorphic and recursive types*, March 1988
34 A. Visser, *A Descending Hierarchy of Reflection Principles*, April 1988
35 F.P.J.M. Voorbraak, *A computationally efficient approximation of Dempster-Shafer theory*, April 1988
36 C. Smoryński, *Arithmetic Analogues of McAloon's Unique Rosser Sentences*, April 1988

37 P.H. Rodenburg, F.J. van der Linden, *Manufacturing a cartesian closed category with exactly two objects*, May 1988

38 P.H. Rodenburg, J.L.M.Vrancken, *Parallel object-oriented term rewriting : The Booleans*, July 1988

39 D. de Jongh, L. Hendriks, G.R. Renardel de Lavalette, *Computations in fragments of intuitionistic propositional logic*, July 1988

40 A. Visser, *Interpretability Logic*, September 1988

41 M. Doorman, *The existence property in the presence of function symbols*, October 1988

42 F. Voorbraak, *On the justification of Dempster's rule of combination*, December 1988

43 A. Visser, *An inside view of EXP, or: The closed fragment of the provability logic of* $I\Delta_0+\Omega_1$, February 1989

44 D.H.J. de Jongh & A. Visser, *Explicit Fixed Points in Interpretability Logic*, March 1989

45 S. van Denneheuvel & G.R. Renardel de Lavalette, *Normalisation of database expressions involving calculations*, March 1989

46 M.F.J. Drossaers, *A Perceptron Network Theorem Prover for the Propositional Calculus*, July 1989

47 A. Visser, *The Formalization of Interpretability*, August 1989

48 J.L.M. Vrancken, *Parallel Object Oriented Term Rewriting : a first implementation in Pool2*, September 1989

49 G.R. Renardel de Lavalette, *Choice in applicative theories*, September 1989

50 C.P.J. Koymans & G.R. Renardel de Lavalette, *Inductive definitions in COLD-K*, September 1989

51 F. Voorbraak, *Conditionals, probability, and belief revision (preliminary version)*, October 1989

52 A. Visser, *On the* Σ^0_1*-Conservativity of* Σ^0_1*-Completeness*, October 1989

53 G.R. Renardel de Lavalette, *Counterexamples in applicative theories with choice*, January 1990

54 D. van Dalen, *L.E.J. Brouwer. Wiskundige en Mysticus*, June 1990

55 F. Voorbraak, *The logic of objective knowledge and rational belief*, September 1990

56 J.L.M. Vrancken, *Reflections on Parallel and Functional Languages*, September 1990

57 A. Visser, *An inside view of EXP, or: The closed fragment of the provability logic of* $I\Delta_0+\Omega_1$, revised version with new appendices, October 1990

58 S. van Denneheuvel, K. Kwast, G.R. Renardel de Lavalette, E. Spaan, *Query optimization using rewrite rules*, October 1990

59 G.R. Renardel de Lavalette, *Strictness analysis via abstract interpretation for recursively defined types*, October 1990

60 C.F.M. Vermeulen, *Sequence Semantics for Dynamic Predicate Logic*, January 1991

61 M.B. Kalsbeek, *Towards the Interpretability Logic of* $I\Delta_0+EXP$, January 1991.

62 D. van Dalen, *I<R, Some Intuitionistic Elementary Equivalences*, February 1991.

63 M. Bezem, *Strong termination of Logic Programs*, March 1991.

64 A. Visser, *The Unprovability of Small Inconsistency*, March 1991.

65 C.M. Jonker, *On the Semantics of Conflict Resolution in Truth Maintenance Systems*, March 1991.

66 F. Voorbraak, *A Preferential Model Semantics for Default Logic*, July 1991.

67 M. Kracht, *Splittings and the finite model property*, October 1991.

68 M. Bezem, *Impredicative recursion: terms depending on order types (extended abstract)*, November 1991.

69 E. Barendsen, M. Bezem, *Bar recursion versus polymorphism (extended abstract)*, December 1991.