Recursion does not always help

Gordon Plotkin
University of Edinburgh
gdp@inf.ed.ac.uk

Abstract

We show that adding recursion does not increase the total functions definable in the typed $\lambda\beta\eta$-calculus or the, possibly partial, functions definable in the $\lambda\Omega$-calculus. As a consequence, adding recursion does not increase the class of partial or total definable functions on free algebras and so, in particular, on the natural numbers.

1 Introduction

As is well known, using Church numerals as "codes" for natural numbers, all partial recursive functions can be defined in the untyped λ-calculus. The definitions make use of recursion. If we switch to a typed framework, the situation changes drastically: in the typed $\lambda\beta\eta$-calculus (or the $\lambda\beta$-calculus) only the extended polynomials can be defined using Church numerals of type $(o \to o) \to (o \to o)$ to represent the set of natural numbers, see [6, 7].

Further functions can be defined if, more generally, one uses representing types of the form $(\sigma \to \sigma) \to (\sigma \to \sigma)$. Fortune et al [4] showed that this is possible non-uniformly, that is, if one uses different such types for the arguments and results. For example, predecessor can be defined non-uniformly, but, as shown by Zakrzewski [11], cannot be defined uniformly. Zakrzewski further showed—contrary to the then common belief—that there are uniform examples. In particular he showed that, for $l \geq 2$, the function

$$f(m, n_0, \ldots, n_{l-1}) = n_{m \bmod l}$$

is uniformly definable, as is, for $l \geq 1$, the characteristic function of the predicate $\leq l$. He further conjectured that adding these to the schema for the extended polynomials characterises all the uniformly definable functions. But, in any case, one cannot so represent all total recursive functions. It is therefore natural to ask which, possibly partial, numerical functions can be defined if one adds recursion at all types, i.e., if one uses the λY-calculus [9].

More generally than numerical functions, free algebras can be represented in the $\lambda\beta\eta$-calculus and then functions on the free algebras can be defined, see [2, p. 38]. So one can again ask whether adding recursion increases the class of definable functions. More generally still, given representations of non-empty sets X_i. $(i = 1, \ldots, k)$, and X and corresponding coding functions, one can ask if adding recursion changes the set of (possibly partial) definable functions $f : X_1 \times \ldots \times X_k \rightharpoonup X$.

Perhaps surprisingly, it turns out (Theorem 1) that no more total functions can be defined in the λY-calculus than in the $\lambda\beta\eta$-calculus, and, further, that no more, possibly partial, functions can be defined in the λY-calculus than in the $\lambda\Omega$-calculus (the $\lambda\beta\eta$-calculus extended with a ground type constant Ω for "undefined"). So, in particular, adding recursion does not increase the available definable partial or total functions on free algebras. The reason is that it is not possible to make use of unbounded recursion depth in the λY-calculus.

The main tool we use to prove our results is due to Werner Damm [3]. It is a kind of "higher-order flow analysis" using a finite domain. It can also be used to prove results of Statman, that the termination of closed λY-terms is decidable, as is whether they have a head normal form.

It is worth remarking that, in contrast, in the second-order λ-calculus where the type of the numerals is $\forall X.(X \to X) \to (X \to X)$, all functions provably total in second-order Peano arithmetic, and so all primitive recursive functions, are definable, see [8, 5]. If recursion is added, all total recursive functions become definable.

This paper is written in honour of Jonathan Seldin on the occasion of his 80th birthday.

2 Definable functions

The typed $\lambda\beta\eta$-calculus is as in [1], say, with types built up from a single ground type o, and, following Church, with variables x^σ carrying their own types. We may omit the types when they can be understood from the context. We write types of the form $\sigma_1 \to \ldots \to \sigma_n \to o$ as $(\sigma_1, \ldots, \sigma_n)$ and say they are n-ary. We write Λ_σ for the set of closed terms of type σ.

By an *extension* λ^+ of the $\lambda\beta\eta$-calculus, we mean a calculus whose terms are λ-terms with additional constants, and with a type-respecting conversion relation between those terms, written:

$$\vdash_{\lambda^+} M = N$$

that is an equivalence relation closed under λ-abstraction, application, and substitution, and which contains $\lambda\beta\eta$-conversion.

Such a calculus λ_2^+ is an *extension* of another such calculus λ_1^+ if the constants of λ_1^+ are also constants of λ_2^+ and, for any λ_1^+- terms M and N, we have:

$$\vdash_{\lambda_1^+} M = N \implies \vdash_{\lambda_2^+} M = N$$

The extension is *conservative* if, for any two such terms, the reverse implication also holds.

A *coding* function for a set X is simply a function from X to some Λ_σ (σ is the representing type). Let λ^+ be an extension of the $\lambda\beta\eta$-calculus. Then, for nonempty sets X_i ($i = 1, \ldots, k$) and X, given coding functions $\gamma_i : X_i \to \Lambda_{\sigma_i}$ ($i = 1, \ldots, k$), and $\gamma : X \to \Lambda_\sigma$, a closed term $F : \sigma_1 \to \ldots \to \sigma_k \to \sigma$ is said to $(\gamma_1, \ldots, \gamma_k, \gamma)$-*define* a (possibly partial) function $f : X_1 \times \ldots \times X_k \rightharpoonup X$ in λ^+ if the following two conditions hold:

$$f(x_1, \ldots, x_k) = x \implies \vdash_{\lambda^+} F\gamma(x_1) \ldots \gamma(x_k) = \gamma(x) \qquad \text{(Graph)}$$

and

$$f(x_1, \ldots, x_k)\uparrow \implies \nexists N \in \Lambda_\sigma.\ \vdash_{\lambda^+} F\gamma(x_1) \ldots \gamma(x_k) = N \qquad \text{(Undef)}$$

If f is total this amounts to

$$\vdash_{\lambda^+} F\gamma(x_1) \ldots \gamma(x_k) = \gamma(f(x_1, \ldots, x_k))$$

If $\gamma_1 = \ldots = \gamma_k = \gamma$, we say that F γ-*defines* f and that f is *uniformly* definable. Note that, with this definition, only total functions can be defined in $\lambda\beta\eta$.

Algebraic datatypes provide examples, as explained in [2, p. 38]. There free algebras $\mathcal{A}_\Sigma$ over given signatures Σ are considered, and *standard representation* types ρ_Σ and coding functions $\gamma_\Sigma : \mathcal{A}_\Sigma \to \rho_\Sigma$ are given. The nontrivial case (the one where the free algebra is nonempty) is where the signature includes at least one constant; in that case the representation type is inhabited. In the case of the natural numbers, the signature Σ_{Nat} is that of one unary function and one constant, the standard representation type is $(o \to o) \to (o \to o)$, and the Church numerals provide the standard coding function:

$$\gamma_{\Sigma_{\mathrm{Nat}}}(m) =_{\mathrm{def}} \lambda f^{o\to o}.\lambda x^o.f^m(x)$$

The free algebra functions and combinations of them using certain representable discriminators are shown standardly definable. Zaionc [10] has given an exact characterisation of the standardly definable functions in terms of a certain limited recursion scheme, as well as more explicit characterisations for the cases of trees and words (this last generalising that for the natural numbers).

One can proceed somewhat more generally. One uses instances $\rho_{\Sigma,\sigma} =_{\mathrm{def}} \rho_\Sigma[\sigma/o]$ of the standard representation type, and one then obtains corresponding coding functions $\gamma_{\Sigma,\sigma} : \mathbb{A}_\Sigma \to \rho_{\Sigma,\sigma}$. We refer to $(\gamma_{\Sigma,\sigma_1}, \ldots, \gamma_{\Sigma,\sigma_k}, \gamma_{\Sigma,\sigma})$-definability (respectively $\gamma_{\Sigma,\sigma}$-definability) as $(\sigma_1, \ldots, \sigma_k, \sigma)$-definability (respectively σ-definability). Taking instances, we note that o-definable functions are σ-definable, for any σ.

Returning to general considerations, let λ_1^+ and λ_2^+ be extensions of the $\lambda\beta\eta$-calculus. Then λ_1^+ and λ_2^+ are *equipotent* for $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable (total) functions, if a (total) function is $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable in λ_1^+ if, and only if, it is in λ_2^+.

The next lemma presents a strategy for showing that all functions definable in one extension of the $\lambda\beta\eta$-calculus are definable in another. It gives requirements on a translation from one calculus to another sufficient to ensure that every function definable in the first is also definable in the second.

Lemma 1. *Let λ_1^+ and λ_2^+ be extensions of the $\lambda\beta\eta$-calculus, and let* tr *be a type and application respecting map from closed λ_1^+-terms to closed λ_2^+-terms which acts as the identity on closed λ-terms. Suppose further that, for every closed λ_1^+-term M and closed λ-term N of the same type we have:*

$$\vdash_{\lambda_1^+} M = N \iff \vdash_{\lambda_2^+} \mathrm{tr}(M) = N \qquad (*)$$

Then every function $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable in λ_1^+ is $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable in λ_2^+. Further, in the case such a function is total, only the implication from left to right in $(*)$ is needed.

Proof. Let $\gamma_i : X_i \to \Lambda_{\sigma_i}$ $(i = 1, \ldots, k)$, and $\gamma : X \to \Lambda_\sigma$ be coding functions, and suppose that the term $F : \sigma_1 \to \ldots \to \sigma_k \to \sigma$ $(\gamma_1, \ldots, \gamma_k, \gamma)$-defines $f : X_1 \times \ldots \times X_k \rightharpoonup X$ in λ_1^+.

Using the assumptions on tr and $(*)$ we have the following implications for the Graph condition.

$$\begin{array}{rcl} f(x_1, \ldots, x_k) = x & \Longrightarrow & \vdash_{\lambda_1} F\gamma_1(x_1)\ldots\gamma_k(x_k) = \gamma(x) \\ & \Longrightarrow & \vdash_{\lambda_2} \mathrm{tr}(F\gamma_1(x_1)\ldots\gamma_k(x_k)) = \gamma(x) \\ & \Longrightarrow & \vdash_{\lambda_2} \mathrm{tr}(F)\gamma_1(x_1)\ldots\gamma_k(x_k) = \gamma(x) \end{array}$$

and, for any $N \in \Lambda_\sigma$, the following implications for the Undef condition:

$$\begin{array}{rcl} \vdash_{\lambda_2} \mathrm{tr}(F)\gamma_1(x_1)\ldots\gamma_k(x_k) = N & \Longrightarrow & \vdash_{\lambda_2} \mathrm{tr}(F\gamma_1(x_1)\ldots\gamma_k(x_k)) = N \\ & \Longrightarrow & \vdash_{\lambda_1} F\gamma_1(x_1)\ldots\gamma_k(x_k) = N \\ & \Longrightarrow & f(x_1, \ldots, x_k)\downarrow \end{array}$$

□

Note that in the case λ_2^+ is a conservative extension of λ_1^+, one can take the translation tr to be the identity (with only an extension being needed in the case of total functions).

3 The $\lambda\Omega^+$-calculus

The $\lambda\Omega^+$-calculus is the $\lambda\beta\eta$-calculus extended with constants $\Omega_\sigma : \sigma$ for every type σ, and the conversion relation generated by β- and η-conversion for all terms. With β-reduction and reverse η-reduction, one has Church-Rosser and long $\beta\eta$-normal forms, just as in the $\lambda\beta\eta$-calculus. Long $\beta\eta$-normal forms containing no Ω_σ are called *proper*, all others are called *improper*. The $\lambda\Omega$-calculus is the subcalculus with just the constant Ω_o (written Ω). The $\lambda\Omega^+$-calculus is a useful intermediary between the λY-calculus and the $\lambda\Omega$-calculus: the Ω_σ act as variables of type σ which can be substituted for to link with those calculi.

The $\lambda\Omega^+$-calculus is a conservative extension of the $\lambda\Omega$-calculus, and, in turn, the $\lambda\Omega$-calculus is a conservative extension of the $\lambda\beta\eta$-calculus. By the remark after Lemma 1 it follows that every function definable in the $\lambda\beta\eta$-calculus is definable in the $\lambda\Omega$-calculus, and that every function definable in the $\lambda\Omega$-calculus is definable in the $\lambda\Omega^+$-calculus (all with the same coding scheme).

Lemma 2. *The $\lambda\beta\eta$ and $\lambda\Omega$ calculi are* equipotent *for $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable total functions.*

Proof. We have already seen that every total function $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable in $\lambda\beta\eta$ is $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable in $\lambda\Omega$. Conversely, suppose that $F : \sigma_1 \to \ldots \to \sigma_k \to \sigma$ $(\gamma_1, \ldots, \gamma_k, \gamma)$-defines a total function f in the $\lambda\Omega$-calculus. The long $\beta\eta$-normal form of F has the form

$$F[\Omega] = \lambda x_1^{\sigma_1} \ldots x_k^{\sigma_k}.\lambda y_1^{\tau_1} \ldots y_l^{\tau_l}.M[\Omega]$$

with $N[\Omega] : o$, where $\sigma = (\tau_1, \ldots, \tau_l)$. Choosing $a \in X$ (recall that X is non-empty) we obtain a closed term $A =_{\text{def}} \gamma(a) : \sigma$

Then

$$F[Ay_1^{\tau_1} \ldots y_l^{\tau_l}] = \lambda x_1^{\sigma_1} \ldots x_k^{\sigma_k}.\lambda y_1^{\tau_1} \ldots y_l^{\tau_l}.M[Ay_1^{\tau_1} \ldots y_l^{\tau_l}]$$

is a λ-term defining the same function, as the function is total. and

$$\vdash_{\lambda\Omega} F[\Omega]\gamma_1(a_1) \ldots \gamma_1(a_k) = \gamma(f(a_1, \ldots, a_k))$$

implies

$$\vdash_{\lambda\beta\eta} F[Ay_1^{\tau_1} \ldots y_l^{\tau_l}]\gamma_1(a_1) \ldots \gamma_1(a_k) = \gamma(f(a_1, \ldots, a_k))$$

□

There is a natural translation from $\lambda\Omega^+$ to $\lambda\Omega$. For n-ary σ, set $\overline{\Omega}_\sigma = \lambda x_1 \dots x_n.\Omega$. Then, for any $\lambda\Omega^+$-term M, let $\overline{M}$ be the $\lambda\Omega$-term obtained from M by replacing all the Ω_σ's in M by $\overline{\Omega}_\sigma$'s.

Lemma 3. *For any $\lambda\Omega^+$-term M, and any λ-term N of the same type we have:*

$$\vdash_{\lambda\Omega^+} M = N \implies \vdash_{\lambda\Omega} \overline{M} = N$$

Proof. As the Ω_σ act as variables, one can replace the Ω_σ in M and N by the corresponding $\overline{\Omega}_\sigma$. As this changes M to $\overline{M}$ and leaves N alone, the result follows. □

It would be interesting to have a syntactic proof of the converse of this lemma; a flow-analyis proof is given below.

4 The λY-calculus

The λY-calculus [9] is the $\lambda\beta\eta$-calculus extended with recursion operators, i.e., constants

$$\mathrm{Y}_\sigma : (\sigma \to \sigma) \to \sigma$$

and conversions $\mathrm{Y}_\sigma F = F(\mathrm{Y}_\sigma f)$, for $F : \sigma \to \sigma$. With reduction rules β, η, and $\mathrm{Y}_\sigma \to \lambda f.f(\mathrm{Y}_\sigma f)$, it is Church-Rosser (see [9] and [2, p. 314]).

There is a natural translation from $\lambda\Omega^+$ to λY. Set $\widehat{\Omega}_\sigma = \mathrm{Y}_\sigma(\lambda x.\, x)$. Then, for any $\lambda\Omega^+$-term M, let $\widehat{M}$ be the λY-term obtained from M by replacing all the Ω_σ's in M by $\widehat{\Omega}_\sigma$'s. We evidently have:

Lemma 4. *For any $\lambda\Omega^+$-term M, and any λ-term N of the same type we have:*

$$\vdash_{\lambda\Omega^+} M = N \implies \vdash_{\lambda \mathrm{Y}} \widehat{M} = N$$

As in the case of Lemma 3, the converse will be obtained by flow-analysis.

In the other direction, we work with particular approximations to Y in the next section. We first consider such approximations in general. Define $\lambda\Omega^+$-terms $\widetilde{\mathrm{Y}}^{(n)}_\sigma : (\sigma \to \sigma) \to \sigma$ by $\widetilde{\mathrm{Y}}^{(n)}_\sigma = \lambda f.f^n(\Omega_{\sigma\to\sigma} f)$ and λY-terms $\mathrm{Y}^{(n)}_\sigma : (\sigma \to \sigma) \to \sigma$ by $\mathrm{Y}^{(n)}_\sigma = \lambda f.f^n(\mathrm{Y}_\sigma(f))$. Then, for any λY-term $M[\mathrm{Y}_{\sigma_1}, \dots, \mathrm{Y}_{\sigma_k}]$ and any $n_1, \dots, n_k$, let $M^{(n_1,\dots,n_k)}$ be the $\lambda\Omega^+$-term $M[\widetilde{\mathrm{Y}}^{(n_1)}_{\sigma_1}, \dots, \widetilde{\mathrm{Y}}^{(n_k)}_{\sigma_n}]$.

Lemma 5. *For any λY-term M, and any λ-term N of the same type we have:*

$$\vdash_{\lambda\Omega^+} M^{(n_1,\ldots,n_k)} = N \implies \vdash_{\lambda \mathrm{Y}} M = N$$

Proof. We have $\vdash_{\lambda\Omega^+} M[\widetilde{\mathrm{Y}}^{(n_1)}_{\sigma_1},\ldots,\widetilde{\mathrm{Y}}^{(n_k)}_{\sigma_k}] = N$. As the $\Omega_{\sigma\to\sigma}$ act as variables, and do not occur in N, $\vdash_{\lambda\mathrm{Y}} M[\mathrm{Y}^{(n_1)}_{\sigma_1},\ldots,\mathrm{Y}^{(n_k)}_{\sigma_k}] = N$ (replacing $\Omega_{\sigma\to\sigma}$'s by Y_σ's). The conclusion follows, as, for any σ and n, we have $\vdash_{\lambda\mathrm{Y}} \mathrm{Y}^{(n)}_\sigma = \mathrm{Y}_\sigma$.

□

We remark that if a version of this lemma for the $\lambda\Omega$-calculus were available, the $\lambda\Omega^+$-calculus would not be needed.

5 Semantics

We work over the simple type hierarchy $\mathcal{O}_\sigma$ of continuous functions starting with Sierpiński space: $\mathcal{O}_o = \mathbb{O}$ (i.e., $\{\bot,\top\}$, with $\bot\leq\top$). As these domains are all finite, this is also the hierarchy of monotone functions. This gives an interpretation of the typed $\lambda\beta\eta$-calculus in a standard way. We write $\mathcal{O}[\![M]\!](\rho)$ for the interpretation of a term M in environment ρ, and generally omit the ρ when M is closed. The interpretation is extended to the $\lambda\Omega^+$-calculus by taking $\mathcal{O}[\![\Omega_\sigma]\!]$ to be $\bot_{\mathcal{O}_\sigma}$ and to the λY-calculus by taking $\mathcal{O}[\![\mathrm{Y}_\sigma]\!]$ to be the least fixed point operator $f \mapsto \bigvee_n f^n(\bot_{\mathcal{O}_\sigma})$.

We have $\mathcal{O}[\![\Omega_\sigma]\!] = \mathcal{O}[\![\overline{\Omega}_\sigma]\!] = \mathcal{O}[\![\widehat{\Omega}_\sigma]\!]$. So, for any closed $\lambda\Omega^+$-term M, we have: $\mathcal{O}[\![M]\!] = \mathcal{O}[\![\overline{M}]\!] = \mathcal{O}[\![\widehat{M}]\!]$. Next, taking $h(\sigma)$ to be the height of the longest ascending chain in $\mathcal{O}_\sigma$, the least-fixed-point operator is the same as the truncated operator $f \mapsto f^{h(\sigma)}(\bot_{\mathcal{O}_\sigma})$, and that is precisely $\mathcal{O}[\![\widetilde{\mathrm{Y}}^{(h(\sigma))}_\sigma]\!]$.

We can now define the translation from λY to $\lambda\Omega^+$. For any closed λY-term $M[Y_{\sigma_1}\ldots,Y_{\sigma_k}]$, set $\widetilde{M} = M^{((h(\sigma_1),\ldots,(h(\sigma_k))}$. Note that we then have: $\mathcal{O}[\![M]\!] = \mathcal{O}[\![\widetilde{M}]\!]$.

As we shall see, $\widetilde{M}$ encodes the maximum recursion depth available to the recursion operators in M. To show this, as mentioned in the introduction, we employ Werner Damm's higher-order flow analysis over a finite model. The analysis is carried out using certain "test functions"

$t_\sigma : \mathcal{O}_\sigma \to \mathbb{O}$. As shown in Lemma 6 below, these test functions distinguish proper from improper normal forms.

The t_σ are defined mutually inductively with $s_\sigma \in \mathcal{O}_\sigma$ by setting:

$$t_\sigma(f) = f s_{\sigma_1} \dots s_{\sigma_n} \qquad\qquad s_\sigma f_1 \dots f_n = \bigwedge_i t_{\sigma_i}(f_i)$$

for $\sigma = (\sigma_1, \dots, \sigma_n)$. Note that, in particular, $t_o(x) = x$ and $s_o = \top$; we further have $t_{\sigma\to\tau} f = t_\tau(f s_\sigma)$.

Lemma 6. *Let $M : \sigma$ be a long $\beta\eta$-normal form in $\lambda\Omega^+$. Then:*

$$t_\sigma(\mathcal{O}[\![M]\!]) = \top \iff M \textit{ is proper}$$

Proof. Let σ be $(\sigma_1, \dots, \sigma_n)$. We proceed by induction. Suppose that M is proper. Then M has the form

$$\lambda f_1 \dots f_n . f_{i_0} M_1 \dots M_k$$

where $\sigma_{i_0} = (\tau_1, \dots, \tau_k)$ and the $N_j =_{\mathrm{def}} \lambda f_1 \dots f_n . M_j$ are strictly smaller closed long $\beta\eta$-normal forms. We now calculate:

$$\begin{array}{rcl}
t_\sigma(\mathcal{O}[\![M]\!]) & = & s_{\sigma_{i_0}}(\mathcal{O}[\![N_1]\!] s_{\sigma_1} \dots s_{\sigma_n}) \dots (\mathcal{O}[\![N_k]\!] s_{\sigma_1} \dots s_{\sigma_n}) \\
& = & \bigwedge_j t_{\tau_j}(\mathcal{O}[\![N_j]\!] s_{\sigma_1} \dots s_{\sigma_n}) \\
& = & \bigwedge_j t_{\sigma_1 \to \dots \to \sigma_n \to \tau_j}(\mathcal{O}[\![N_j]\!]) \quad \text{(by the above remark)} \\
& = & \top \quad \text{(by induction hypothesis)}
\end{array}$$

Suppose instead that M is improper. Then it either has the form:

$$\lambda f_1 \dots f_n . \Omega_{\sigma_{i_0}} M_1 \dots M_k$$

or

$$\lambda f_1 \dots f_n . f_{i_0} M_1 \dots M_k$$

where some $N_j =_{\mathrm{def}} \lambda f_1 \dots f_n . M_j$ is improper (and so, by the induction hypothesis $t_{\sigma_1 \to \dots \to \sigma_n \to \tau_j}(\mathcal{O}[\![N_j]\!]) = \perp$).

In the first case we have

$$\begin{array}{rcl}
t_\sigma(\mathcal{O}[\![M]\!]) & = & \mathcal{O}[\![\Omega_{\sigma_{i_0}}]\!](\mathcal{O}[\![N_1]\!] s_{\sigma_1} \dots s_{\sigma_n}) \dots (\mathcal{O}[\![N_k]\!] s_{\sigma_1} \dots s_{\sigma_n}) \\
& = & \perp
\end{array}$$

In the second we have:

$$\begin{array}{rcl} t_\sigma(\mathcal{O}[\![M]\!]) & = & \bigwedge_j t_{\sigma_1\to\ldots\to\sigma_n\to\tau_j}(\mathcal{O}[\![N_j]\!]) \\ & = & \bot \end{array}$$

□

As a straightforward consequence of Lemma 6 we have:

Lemma 7. *A closed $\lambda\Omega^+$-term $M:\sigma$ is provably equal to a closed λ-term (equivalently, has a proper normal form) if, and only if, $t_\sigma(\mathcal{O}[\![M]\!]) = \top$.*

6 Equipotence

We establish equipotence by using Lemma 7 to show the the various translations between our calculi satisfy the conditions of Lemma 1. We already know that the identity translation from $\lambda\Omega$ to $\lambda\Omega^+$ does. For the remaining three translations we have:

Lemma 8.

1. *For any closed $\lambda\Omega^+$-term M and closed λ-term N of the same type we have:*
$$\vdash_{\lambda\Omega^+} M = N \iff \vdash_{\lambda\Omega} \overline{M} = N$$
2. *For any closed $\lambda\Omega^+$-term M and closed λ-term N of the same type we have:*
$$\vdash_{\lambda\Omega^+} M = N \iff \vdash_{\lambda\mathrm{Y}} \widehat{M} = N$$
3. *For any closed $\lambda\mathrm{Y}$-term M and closed λ-term N of the same type we have:*
$$\vdash_{\lambda\mathrm{Y}} M = N \iff \vdash_{\lambda\Omega^+} \widetilde{M} = N$$

Proof.

1. We already have the implication from left to right by Lemma 3. In the other direction, as $\mathcal{O}[\![M]\!] = \mathcal{O}[\![\overline{M}]\!]$, and as $\vdash_{\lambda\Omega} \overline{M} = N$, we have $t_\sigma(\mathcal{O}[\![M]\!]) = t_\sigma(\mathcal{O}[\![N]\!]) = \top$ by Lemma 7, and so, by the same lemma, $\vdash_{\lambda\Omega^+} M = N'$ for some closed λ-term N'. So then $\vdash_{\lambda\Omega} \overline{M} = N'$ by Lemma 3. But then $\vdash_{\lambda\Omega} N' = N$, as $\vdash_{\lambda\Omega} \overline{M} = N$. So as $\vdash_{\lambda\Omega^+} M = N'$ and as $\lambda\Omega^+$ is an extension of $\lambda\Omega$, we have $\vdash_{\lambda\Omega^+} M = N$ as required.

2. We already have the implication from left to right by Lemma 4. In the other direction, as $\mathcal{O}[\![M]\!] = \mathcal{O}[\![\widehat{M}]\!]$, and as $\vdash_{\lambda\mathrm{Y}} \widehat{M} = N$, arguing as in the previous case, employing Lemma 7, we find that $\vdash_{\lambda\Omega^+} M = N'$ for some closed λ-term N'. So then $\vdash_{\lambda\mathrm{Y}} \widehat{M} = N'$ by Lemma 3. But then $\vdash_{\lambda\mathrm{Y}} N' = N$, as $\vdash_{\lambda\mathrm{Y}} \widehat{M} = N$. So, as $\lambda\mathrm{Y}$ is conservative over $\lambda\beta\eta$, and, as $\vdash_{\lambda\Omega^+} M = N'$, we have $\vdash_{\lambda\Omega^+} M = N$ as required.

3. We have the implication from right to left by Lemma 5. In the other direction as $\mathcal{O}[\![\widetilde{M}]\!] = \mathcal{O}[\![M]\!] = \mathcal{O}[\![N]\!]$, we have $\vdash_{\lambda\Omega^+} \widetilde{M} = N'$ for some closed λ-term N' by Lemma 7. So then $\vdash_{\lambda\mathrm{Y}} M = N'$ by Lemma 5. But then $\vdash_{\lambda\mathrm{Y}} N' = N$, as $\vdash_{\lambda\mathrm{Y}} M = N$. So, as $\lambda\mathrm{Y}$ is conservative over $\lambda\beta\eta$, and, as $\vdash_{\lambda\Omega^+} M = N'$, we have $\vdash_{\lambda\Omega^+} M = N$ as required.

$\square$

With this lemma in hand, with the identity translation from $\lambda\Omega$ to $\lambda\Omega^+$, and with Lemma 2 on the equipotence of the $\lambda\beta\eta$- and $\lambda\Omega$-calculi for total functions, we obtain our theorem that recursion does not help:

Theorem 1. *Let $\gamma_i : X_i \to \Lambda_{\sigma_i}$ $(i = 1, \ldots, k)$, and $\gamma : X \to \Lambda_\sigma$ be coding functions. Then:*

1. *The $\lambda\mathrm{Y}$-calculus is equipotent with the $\lambda\Omega$-calculus for $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable functions.*

2. *The $\lambda\mathrm{Y}$-calculus is equipotent with the $\lambda\beta\eta$-calculus for $(\gamma_1, \ldots, \gamma_k, \gamma)$-definable total functions.*

We remark that one might add the converse of the Graph condition, viz:

$$\vdash_{\lambda^+} F\gamma(x_1)\ldots\gamma(x_k) = \gamma(x) \implies f(x_1, \ldots, x_k) = x$$

to the definition of the $(\gamma_1, \ldots, \gamma_k, \gamma)$-definability of a function f in an extension λ^+ of the $\lambda\beta\eta$-calculus by a term F. With this addition, Lemma 1 still goes through. Further, the addition makes no difference to definability provided that the coding function γ is λ^+-*injective* by which we mean:

$$\vdash_{\lambda^+} \gamma(x) = \gamma(y) \implies x = y$$

So, with this addition to the definition of definability, Part 1 of Theorem 1 still holds, and Part 2 holds when γ is $\lambda\beta\eta$-injective.

Some known undefinabilities for natural number functions follow from Theorem 1. As we mentioned, Zakrzewski proved that predecessor is not uniformly definable and Statman proved[1], that neither equality nor inequality ($\leq$) are uniformly definable.

These facts follow from the theorem. For the three functions are interdefinable via suitable recursions, and so, if one of them was uniformly definable, one could uniformly define arbitrary partial recursive functions in the λY-calculus. They also follow from Statman's result [9] that it is decidable whether a λY-term has a $\lambda\beta\eta$ normal form, since that implies that all definable partial recursive functions have recursive domains.

We can use the fact that arbitrary recursion depth is not available to show that even equality to a fixed number may not be σ-definable. Define e_m by:

$$e_m(n) = \begin{cases} 1 & (n = m) \\ 0 & (\text{otherwise}) \end{cases}$$

Fact 1. *For any σ, e_m is not σ-definable for $m > h(\sigma \to \sigma)$.*

Proof. We write ω_σ for $\rho_{\Sigma_{\mathrm{Nat}},\sigma}$, i.e., for $(\sigma \to \sigma) \to (\sigma \to \sigma)$, and $\underline{m}_\sigma$ for $\gamma_{\Sigma_{\mathrm{Nat}},\sigma}(m)$, i.e., for $\lambda f^{\sigma\to\sigma}.\lambda x^\sigma.f^m(x)$. Suppose, for the sake of contradiction, that a term E_m σ-defines e_m for $m > h(\sigma \to \sigma)$. Let $Z = \mathrm{Y}_{\omega_\sigma\to\omega_\sigma}F$ where

$$F = \lambda f x.\mathtt{if}\, E_m x\, \mathtt{then}\, \underline{0}_\sigma\, \mathtt{else}\, f(x + \underline{1}_\sigma)$$

(Recall that the conditional is an extended polynomial.)

We have $\vdash_{\lambda\mathrm{Y}} Z\underline{0}_\sigma = \underline{0}_\sigma$. However, using induction on k, one shows that $\vdash_{\lambda\Omega^+} \widetilde{\mathrm{Y}}_\sigma^{(k)}F\underline{l}_\sigma = \Omega_\sigma$ when $k + l < m$. So $\vdash_{\lambda\Omega^+} \widetilde{\mathrm{Y}}_\sigma^{(h(\sigma\to\sigma))}F\underline{0}_\sigma = \Omega_\sigma$ in particular. Hence, by Lemma 5, we have $\vdash_{\lambda\mathrm{Y}} \mathrm{Y}_\sigma F\underline{0}_\sigma = \Omega_\sigma$. So $\vdash_{\lambda\mathrm{Y}} \underline{0}_\sigma = \Omega_\sigma$ contradicting the consistency of λY. □

It follows that comparisons to a suitably large fixed number are not σ-definable either. Using a similar bounded recursion depth argument

[1]Personal communication, reported in [4]

one can obtain Zakrzewski's result [11] that $\lfloor n/2 \rfloor$ is not uniformly definable.

Using the flow analysis technique we can also prove Statman's decidability result. We have a version of Lemma 7 for the λY-calculus:

Lemma 9. *A closed λY-term $M:\sigma$ is provably equal to a closed λ-term (equivalently, has a $\lambda\beta\eta$ normal form) if, and only if, $t_\sigma(\mathcal{O}[\![M]\!]) = \top$.*

Proof. The implication from left to right follows from Lemma 6. In the other direction, as $\mathcal{O}[\![M]\!] = \mathcal{O}[\![\widetilde{M}]\!]$ and $\widetilde{M}$ has a normal form, the same lemma tells us that $\widetilde{M}$ has a proper normal form. The conclusion follows, by Lemma 5. ☐

Statman's result then follows as it is evidently decidable for closed λY-terms M whether or not $t_\sigma(\mathcal{O}[\![M]\!]) = \top$. We remark that, using a different flow analysis, one can obtain another of Statman's results, that it is decidable whether a given λY-term has a head normal form. One takes:

$$t_\sigma(f) = f s_{\sigma_1} \dots s_{\sigma_n} \qquad\qquad s_\sigma f_1 \dots f_n = \top$$

Acknowledgements

This paper is based on a manuscript I wrote on a visit to MIT in 1982, a visit for which I am grateful to Albert Meyer. However, the manuscript sank below the waves until Paweł Urzyczyn very kindly sent me a copy. I am delighted to thereby have had the opportunity to have something to tell Jonathan.

The paper will appear in *The Mathematical Foundation of Computation*, a Festschrift for Jonathan Seldin on the occasion of his 80th birthday, edited by Fairouz Kamareddine. I am grateful to College Publications for permission to post it on the web, and I am grateful to Pawel for many helpful comments on drafts.

References

[1] Henk Barendregt. *The lambda calculus - its syntax and semantics.* Studies in logic and the foundations of mathematics 103, North-Holland, 1985.

[2] Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda calculus with types.* Cambridge University Press, 2013.

[3] Werner Damm. *The IO-and OI-hierarchies.* Theor. Comput. Sci., 20(2), 95–207, 1982.

[4] Steven Fortune, Daniel Leivant, and Michael O'Donnell. *The expressiveness of simple and second-order type structures.* Journal of the ACM, 30(1), 151–185, 1983.

[5] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types.* Cambridge Tracts in Theoretical Computer Science, Vol. 7., Cambridge University Press, 1989.

[6] Helmut Schwichtenberg. *Definierbare Funktionen im λ-Kalül mit Typen.* Arch. Math. Log., 17(3-4), 113–114, 1975.

[7] Richard Statman, *The Typed lambda-calculus is not elementary recursive.* Theor. Comput. Sci., 9, 73–81, 1979.

[8] Richard Statman. *Number theoretic functions computable by polymorphic programs.* In 22nd Annual Symposium on Foundations of Computer Science, 279–282, IEEE Computer Society, 1981.

[9] Richard Statman. *On the lambda Y calculus.* Ann. Pure Appl. Log., 130(1–3), 325-337, 2004.

[10] Marek Zaionc. *λ-definability on free algebras.* Ann. Pure Appl. Log., 51(3), 279–300, 1991.

[11] Mateusz Zakrzewski. *Definable functions in the simply typed lambda-calculus.* arXiv preprint cs/0701022, 2007.