Available online at www.sciencedirect.com

ScienceDirect

Indagationes Mathematicae 29 (2018) 450–458

indagationes mathematicae

www.elsevier.com/locate/indag

Virtual Special Issue - L.E.J. Brouwer after 50 years

Fixed points in lambda calculus. An eccentric survey of problems and solutions

Benedetto Intrigila[a,*], Richard Statman[b]

[a] *Università di Roma "Tor Vergata", Rome, Italy*
[b] *Carnegie-Mellon University, Pittsburgh, PA, USA*

Abstract

The fact that every combinator has a fixed point is at the heart of the λ-calculus as a model of computation. We consider several aspects of such phenomenon; our specific, perhaps eccentric, point of view focuses on problems and results that we consider worthy of further investigations. We first consider the relation with *self application*, in comparison with the opposite view, which stresses the role of *coding*, unifying the first and the second fixed point theorems. Then, we consider the relation with the *diagonal argument*, a relation which is at the origin of the fixed point theorem itself. We also review the *Recursion Theorem*, which is considered a recursion theoretic version of the fixed point theorem. We end considering *systems of equations* which are related to fixed points.

1. Fixed points and self-application

One of the most important features of λ-calculus is the fact that every combinator (i.e. closed term) F has a fixed point X. Let F be any combinator and $H \equiv \lambda x.F(xx)$, then $HH \equiv (\lambda x.F(xx))H =_{\lambda\beta} F(HH)$ and therefore $X \equiv HH$ is a fixed point for F.

As observed in [2], from a computational point of view the striking difference with other fixed point theorems, such as the Banach Fixed Point Theorem or the Brouwer Fixed Point Theorem, is that the computation starts from X to arrive to FX.

A very important question is to explain such phenomenon. The literature on this subject is immense, and we only consider some specific aspects.

* Corresponding author.
E-mail address: intrigil@mat.uniroma2.it (B. Intrigila).

http://dx.doi.org/10.1016/j.indag.2017.06.003

A possible approach is to consider *self-application* as the key feature behind the existence of fixed points. This point of view is adopted in [2], where it is also pointed out that some kinds of self application are implicit in Gödel sentences and in Kleene Recursion Theorem.

To explain this point, assume that one looks for a fixed point of the form ZZ for F. Then, $F(ZZ) =_{\lambda\beta} ZZ$ and, by abstraction, $(\lambda x.F(xx))Z =_{\lambda\beta} ZZ$. Then, $Z \equiv \lambda x.F(xx)$ by *pattern matching*. Actually, this seems not far from the actual process leading H.B. Curry from Russell Paradox to the fixed point theorem for λ-calculus (see [7] and [1]).

Curry formulated the Paradox as follows. Instead of $Z \in P$ write PZ, transforming set theoretic membership into the predicate P being affirmed of the subject Z, and write Neg for logical negation. Russell Paradox takes then the form:

$$ZZ \equiv Neg(ZZ).$$

Curry observed that, in this form, the Paradox asks for a *a fixed point* for the negation operator, which is impossible in classical logic. (In [11], Skolem remarked that this is possible in Lukasievich Logic; one gets the value $1/2$, that is *maximum uncertainty*.)

On the other hand, as remarked above, such formulation displays the right pattern to find a fixed point for *every term*, when self application is available.

So, in λ-calculus the paradox is avoided as a fixed point always exists; therefore, not only nothing similar to the classical negation operator Neg can exist, but also negation cannot be simulated by a function that systematically alters its input. In *Set Theory*, the paradox is avoided by a completely different mechanism, that is by requiring that some classes are *too big* to exist (Zermelo) or to be properly treated as sets (Von Neumann, Bernays and others).

With respect to this point of view, one can see the fixed point theorem also as *a restrictive principle*. As an example, consider a function F defined as follows:

$$FXYWZ = W \text{ if } X = Y;$$
$$FXYWZ = Z \text{ otherwise.}$$

We can observe that $\lambda x.Fx\underline{1}\ \underline{0}\ \underline{1}$ has not a fixed point and we conclude that such a function cannot exist in λ-calculus.

Therefore, the difficult task of separating admissible and not admissible collections has been replaced by the equally difficult task of separating admissible and not admissible functional behaviors.

2. Self-application or application to codes?

Another natural possibility is to assume that some kind of *coding* is the crucial property. From this point of view, what actually takes place is that a function is applied to some code of the function itself.

Outside λ-calculus, we find such approach also in *programming languages*, where the instructions which define the function are used as a code of the function, and to take the function as an argument actually means to have a reference to such instructions (e.g. in programming language *C*, where, however, the typing mechanism imposes several restrictions). This idea is also used in some *operational semantics* approaches.

Coming back to λ-calculus, we want to point out which formal properties are required on *codes* to ensure the existence of a fixed point. To do this, we need some definitions.

Assume that a set of closed terms $\mathcal{C}$ has been fixed. We call the elements of $\mathcal{C}$ *codes*. Assume moreover that the following exists:

- a decoding term **Decod** such that for every M there is a N in $\mathcal{C}$ such that **Decod** $N =_{\lambda\beta} M$ (every term has a code);
- a term **App** such that for every P, Q in $\mathcal{C}$, **App** $P\ Q =_{\lambda\beta} N$ for some N in $\mathcal{C}$ such that **Decod** $N =_{\lambda\beta}$ **Decod** P (**Decod** Q) (application can be coded);
- an inverse of **Decod**, that we call **Cod**, on codes: for every N in $\mathcal{C}$ **Cod** $N =_{\lambda\beta} P$ for some P in $\mathcal{C}$ such that **Decod** $P =_{\lambda\beta} N$.

Now, given such coding framework, we have the following fixed point theorem:

Proposition 1. *For every M there exists an $N \in \mathcal{C}$ such that* **Decod** $N =_{\lambda\beta} MN$.

Proof. Define P to be $\lambda x.M(\mathbf{App}\ x(\mathbf{Cod}\ x))$. Now, P has a code U and then

$$PU =_{\lambda\beta} M(\mathbf{App}\ U(\mathbf{Cod}\ U))$$

but **Cod** $U =_{\lambda\beta} Q$ for some Q in $\mathcal{C}$ such that

$$\mathbf{Decod}\ Q =_{\lambda\beta} U$$

and **App** $U\,Q =_{\lambda\beta} N$ for some N in $\mathcal{C}$ such that

$$\mathbf{Decod}\ N =_{\lambda\beta} \mathbf{Decod}\ U\ (\mathbf{Decod}\ Q) =_{\lambda\beta} PU.$$

Therefore, **Decod** $N =_{\lambda\beta} MN$.

It is easy and left to the reader to see that the second fixed point theorem (see [2] 6.5.9) is a particular case of the setting above. This holds also for the first fixed point theorem, by putting $\mathcal{C}$ to be the set of all terms and **Decod** $=$ **Cod** $=$ **App** $=$ **I**.

The fixed point theorem is therefore interpreted as an *abstraction* on the specific coding mechanism, by letting the term to be a code for itself.

It is useful to remark that we have considered only the formal properties of codes which ensure the existence of a fixed point in the sense specified by Proposition 1. A natural and related question is concerned with a full encoding of the *syntax* of lambda terms. Also in this case, it is possible to abstract the basic properties of an adequate coding system, see [8].

We end this section by briefly touching another approach: what properties of types insure the existence of fixed points. For Y to be a fixed point operator, Y must satisfy the following equation:

$$Y = \lambda f.f(Yf)$$

which simple types with $Y : (p \to p) \to p$ for $f : p \to p$. Curry's fixed point operator

$$\mathbf{Y}_C = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

has this type for x with the recursive type:

$$p = p \to p.$$

Much more can be said about this and we refer the reader to part II of [3].

3. Fixed points and diagonalization

In the theory of recursive functions, as it is well-known, *diagonalization* shows that some recursive functions must be *partial functions*. In λ-calculus, diagonalization shows that every combinator must have a *fixed point*.

To see this, let us enumerate the λ-terms (inside λ-calculus) as follows:

$$\{\mathbf{E}\underline{n}\ \underline{m} |\ n, m \in \mathbb{N}\}$$

where $\mathbf{E}$ is Barendregt universal enumerator (see [2] Theorem 8.1.6) and we adopt the usual convention that $\underline{n}$ represents the nth Church numeral.

Now, let M be any combinator. Then, the diagonalization $\lambda x.M(\mathbf{E}xx)$ must have some index n_0 and therefore

$$\mathbf{E}\underline{n}_0\ \underline{n}_0 = \lambda x.M(\mathbf{E}xx)\ \underline{n}_0 = M(\mathbf{E}\underline{n}_0\ \underline{n}_0).$$

A similar connection between fixed points and diagonalization can be found in Category Theory, *via* the Lawvere fixed point theorem [6].

This theorem states that in a cartesian closed category, if there is a morphism $A \to B^A$ which is point-surjective (meaning that $\hom(1, A) \to \hom(1, B^A)$ is surjective), then every endomorphism of B has a fixed point (meaning a morphism $1 \to B$ which is fixed by the endomorphism).

In set theoretic terms, if there is surjective function that maps A onto B^A, then every function $f : B \to B$ has a fixed point. This however is classically possible only in trivial cases.

Here, we propose a version of Lawvere Theorem which appears to be suitable for λ-calculus.

Let the sets A and B be fixed. We consider *typed terms* to specify functions involving A and B (considered as types).

In particular, we have

- variables $x_1^A, x_2^A, x_3^A, \ldots$ of type A;
- variables $x_1^B, x_2^B, x_3^B, \ldots$ of type B;
- constants $\mathbf{a}, \mathbf{a}', \mathbf{a}'', \ldots$ of type A, for every element of A;
- constants $\mathbf{b}, \mathbf{b}', \mathbf{b}'', \ldots$ of type B, for every element of B;
- a constant $\mathbf{e}$ of type $A \to (A \to B)$;
- typed λ-terms constructed from the basic terms specified in the previous items, by λ-abstraction on variables of type A and B and by typed application.

We call such terms λ-**e**-terms. It is clear that once a function $e : A \to B^A$ is given then the closed λ-**e**-terms, with functional type, determine a well-defined class of functions. By abuse of terminology, we call such functions the λ-**e**-*functions* and denote by $\|F\|$ the function corresponding to the closed term F.

Proposition 2. *Assume that $e : A \to B^A$ is such that evaluating* **e** *as e, then every* λ-**e**-*function of type $A \to B$ is in the range of e. Then, every* λ-**e**-*function of type $B \to B$ has a fixed point.*

Proof. Let f be a λ-**e**-function of type $B \to B$, and let F be the term defining f. Then, the term $G \equiv \lambda x^A.F(\mathbf{e}x^A x^A)$, specifies a well-defined function g of type $A \to B$. By our assumptions, g is the range of e. So, let a be such that $e(a) = g$. Now, we have

$$\begin{aligned} e(a)(a) &= \|G\mathbf{a}\| && \text{(since } G \text{ represents } g\text{)} \\ &= \|(\lambda x^A.F(\mathbf{e}x^A x^A))\mathbf{a}\| && \text{(by the definition of } G\text{)} \\ &= \|F(\mathbf{eaa})\| && \text{(by } \beta\text{-reduction)} \\ &= f(e(a)(a)) && \text{(since } F \text{ represents } f\text{)} \end{aligned}$$

and this ends the proof.

Observe that, to obtain a fixed point from the diagonal argument, we have to pay the price of a *circular construction*, stipulating the existence of an "enumeration function" e which has in its range functions which are defined in terms of e itself.

Now, λ-terms also behave as functions, so that we can recast in this setting the construction given at the beginning of the present section. To be more precise, let $\boldsymbol{\mathcal{M}_0}$ denote the set of closed λ-terms modulo β-convertibility (the so called *Term Model*, see [3] pag. 111); we assign to the constant **e** the Barendregt universal enumerator **E** interpreted (modulo β-convertibility) as a function $e_{\mathbf{E}}$ of type $\mathbb{N} \to \boldsymbol{\mathcal{M}_0}^{\mathbb{N}}$. Indeed to every natural number n, **E** assigns a λ-term $\mathbf{E}\underline{n}$ which, in turn, assigns to every natural number m, the element of $\mathcal{M}_0$ corresponding to $\mathbf{E}\underline{n}\,\underline{m}$. It is immediate that every λ-**e**-function of type $A \to B$ is in the range of $e_{\mathbf{E}}$.

Also, it is easy to obtain directly the fixed point theorem for λ-calculus, by setting $A = B = \boldsymbol{\mathcal{M}_0}$. In this case, we assign to the constant **e** the identity function, interpreted as a function that to every λ-term (modulo β-convertibility) assigns the function $\boldsymbol{\mathcal{M}_0} \to \boldsymbol{\mathcal{M}_0}$ corresponding to the λ-term itself.

Outside λ-calculus, we show in the following Section that also the *Recursion Theorem* can be interpreted in this way.

3.1. Brouwer fixed point theorem

It seems appropriate to discuss whether is possible to obtain the Brouwer Fixed Point Theorem in this setting. (Observe that, as far as we know, it is an open problem to obtain the Brouwer Fixed Point Theorem from the Lawvere Fixed Point Theorem (see [9])).

Consider the simplest case of the theorem, which states that every continuous function $f : [0, 1] \to [0, 1]$ has a fixed point.

As it is well known, it is possible *to code* every continuous function of the given type, by a real number. Therefore, by letting A be the set of all code numbers, and $B = [0, 1]$ we can find a map e such that to every real number in $[0, 1]$, which is a code, assigns a continuous function $[0, 1] \to [0, 1]$ and therefore a function from A to B. Moreover, every continuous function has a code. To conclude that every continuous function has a fixed point, we must ensure that also other functions have a code. E.g. letting a be any real number in $[0, 1]$, which is a code, the function defined by the term $\lambda x^A.\mathbf{e}x^A\mathbf{a}$ must have a code.

The simplest solution would be that codes have a *continuous behavior*, that is if $\{c_n\}_{n\in\mathbb{N}}$ is a sequence of codes convergent to code c then

$$\lim_{c_n \to c} e(c_n) = e(c)$$

where the limit is taken w.r.t. some topology in the functional space $\mathcal{C}[0, 1]$.

It is not clear how to obtain a coding with this good property and, moreover, it is possible that such a coding cannot exist at all.

In this context, it is perhaps interesting to note that to make $\mathcal{P}(\omega)$ a model of λ-calculus (the so called *Graph Model* due to Plotkin and Scott), a crucial aspect is the choice of the topology (the so called *Scott Topology*) which allows a coding mechanism which is *continuous* (see [2] 18.1).

In our opinion, to find out a coding able to cope with the geometric aspects of the Brouwer Fixed Point Theorem would be of great interest. Also, a negative answer would be of great interest, as a limit to the coding mechanisms induced by the geometric nature of the involved topological spaces.

4. Fixed points and the recursion theorem

It is well known that the Recursion Theorem, due to Kleene (see [10], pag. 180) is related to the existence of fixed points in λ-calculus.

The analysis of the proof of Kleene, done in [2] (see also [15]), can be summarized as follows:

- given a recursive function ϕ, the basic step is "to make ϕ total", that is to associate to ϕ a total recursive function ψ such that on every code number n, if ϕ is defined on n, $\phi(n)$ and $\psi(n)$ return equivalent codes (i.e. that represent the same partial function);
- once this has been done, self-application (*via codes*) is always defined so that one can apply the λ-calculus machinery to get a fixed point.

The first step is realized by Kleene with an ingenious, purely recursion-theoretic, construction (see [10]). Moreover, this construction is effective, so that one can define a recursive total function t such that given the code n_ϕ of ϕ returns the code n_ψ of ψ. Now, w.l.o.g. we can assume that every natural number is a code number. Therefore, with respect to the setting of the previous paragraph, we can put

- $A = B = \mathbb{N}$
- $e(x) = \phi_{t(x)}$

so that the existence of a fixed point can be derived from Proposition 2. To see this, observe that Kleene construction (the "making total") is not only effective but also uniform. So, as an example, letting n be any natural number, the function defined by the term $\lambda x^A.\mathbf{e}x^A\mathbf{n}$ is effective:

given any natural number m compute $\phi_{t(m)}$ *and return* $\phi_{t(m)}(n)$

and therefore has a code. The other cases can be treated similarly.

On the other hand, it is known that the Recursion Theorem (in a suitable categorical formulation) can be obtained from the Lawvere Fixed Point Theorem, see [16].

Coming back to the construction which "makes total" a recursive function on codes, it is natural to ask if also this step can be performed *via* λ-calculus, where every function is total. In the following we show that this is possible, perhaps suggesting that this issue deserves further investigations.

Let ϕ_n be the nth function in a fixed effective enumeration of partial recursive functions. The following is a version of the Recursion Theorem, called Rogers Fixed Point Theorem (see [10], 11.2 Theorem I)).

Proposition 3. *Let f be a total recursive function; then, there is an n such that $\phi_{f(n)} \cong \phi_n$.*

where $\cong$ is equality between numerical functions.

Proof. By the uniform representability of recursive functions in λ-calculus, there exists a term R such that for every natural number n, $R\underline{n}$ is a term representing the recursive function of index n.

Now, let f be a total recursive function. Given any closed term M consider the following program $\mathcal{P}_M$: for every input n if $M\underline{n}$ is convertible to some Church numeral $\underline{m}$ for some m, then $\mathcal{P}_M(n)$ returns m, and is undefined otherwise. Therefore, $\mathcal{P}_M$ specifies a well defined recursive function of index e_M. Obviously, e_M can be effectively computed from the Gödel number of M, $\lceil M \rceil$, and let g be the recursive function such that $g(\lceil M \rceil) = e_M$.

By the Second Fixed Point Theorem of λ-calculus, let N be a term such that $N =_\beta R(F(G\underline{\lceil N \rceil}))$ where F and G represent f and g, respectively. Now, $e = g(\lceil N \rceil)$ is the required fixed point.

5. Dual fixed point problems

In this section, we consider several problems in λ-calculus that, in a sense are *dual* to fixed point equations. Such problems turn out to be naturally related with representation of semigroups and monoids in λ-calculus, a subject which originated with Church [4].

We start by observing that it is not easy to understand the structure of the set of fixed points of a given non-constant combinator.

Part of the difficulty is the variety of operators which can be applied to construct such fixed points (see [2], page 138, on the Böhm sequence, and also [5]). Therefore, it is natural to consider the dual problem:

$$\{G \mid FG = G\} \leftrightarrow \{F \mid FG = G\}$$

which automatically carries a *semigroup* (with the η-rule, a *monoid*) structure.

In addition, since combinators are both functions and arguments, they can act on one another by both application and composition. This suggests a second semigroup $\{F \mid \mathbf{B}FG = G\}$, where $\mathbf{B}$ is the usual compositor. These are related in the obvious way by the maps:

$$F \mapsto \mathbf{C^*}F.$$

and

$$G \mapsto \mathbf{B}(\mathbf{C^*}G)\mathbf{D}$$

where $\mathbf{C^*}$ is the combinator $\lambda xy.yx$ and $\mathbf{D} \equiv \mathbf{Y}_C(\lambda fxyz.(fx(zy)))$ (recall that $\mathbf{Y}_C$ is the Curry fixed point combinator). Hence,

$$\mathbf{B}(\mathbf{C^*}F)(\mathbf{B}(\mathbf{C^*}G)\mathbf{D}) = \mathbf{B}(\mathbf{C^*}(FG))\mathbf{D}.$$

The map $(F, G) \mapsto (\mathbf{C^*}F, \mathbf{B}(\mathbf{C^*}G)\mathbf{D})$ injects the first monoid into the second one. For it can be shown that for any U, V if $\mathbf{B}(\mathbf{C^*}U)\mathbf{D} = \mathbf{B}(\mathbf{C^*}V)\mathbf{D}$ then $U = V$ (see [14]). So, the original dual is recoverable.

In the following, we shall consider some interesting cases of the above mentioned semigroups and monoids.

Consider an RE set of closed terms $\mathcal{A}$ closed under beta conversion. The members of $\mathcal{A}$ generate a free monoid under the Böhm map:

$$M \mapsto \mathbf{C^*}M.$$

Observe that the generated monoid is free because each element is uniquely of the form $\lambda a.aM_1 \ldots M_m$ (modulo β conversion), with $M_i \in \mathcal{A}$, for $i = 1, \ldots, m$.

Here, we intend to include the Church numeral $\underline{1} =_\eta \mathbf{I}$ as well as the identity $\mathbf{I}$.

Definition 1. If $\mathcal{A}''$ is a set of terms closed under β-conversion, we say that $\mathcal{A}'$ is *fixed-pointswise representable on* $\mathcal{A}''$ if the set $\{L \mid LX = X\ \forall X \in \mathcal{A}''\}$ is equal to the free monoid generated by $\{L \mid \exists M \in \mathcal{A}'\ L = \mathbf{C^*}M\}$.

Note here that we have specifically allowed $\mathcal{A}''$ to contain open terms. We recall some of the definitions of [13] with a few small changes. T is the fixed point combinator of Böhm with a free variable b (see the definition below) and $\mathbf{Y}_C$ is Curry fixed point combinator.

- $\mathbf{E}_\mathcal{A} \equiv$ the enumerator of the set $\{\mathbf{C^*}M \mid M \in \mathcal{A}\}$
- $T \equiv (\lambda xyz.z(xxyz))(\lambda xyz.z(xxyz))b$

- $A' \equiv \lambda fg.\lambda xyz.fx(a(\mathbf{E}_{\mathcal{A}}x))(f(\underline{S}x)y(g(\underline{S}x))z)$
- $A'' \equiv \lambda fg.\lambda x.f(\underline{S}x)(a(\mathbf{E}_{\mathcal{A}}(\underline{S}x))(g(\underline{S}x))(gx))$
- $G \equiv T(\lambda u.A''(T(\lambda v.A'vu))u)$
- $F \equiv T(\lambda u.A'uG)$
- $H \equiv \lambda xa.F\underline{0}(ax)(G\underline{0})$
- $J \equiv \mathbf{Y}_C(\lambda f.\lambda xy.f(x(Hy)))(\mathbf{Y}_C(\lambda g.g(H(\mathbf{E}_{\mathcal{A}}\underline{0}))))$
- $L \equiv \mathbf{Y}_C(\lambda fxy.f(x(Jy)))$
- $P \equiv \mathbf{Y}_C(\lambda f.fJ)$
- $Q \equiv LP$
- $L' \equiv \mathbf{Y}_C(\lambda f.\lambda xy.\langle f, x\rangle)$
- $L'' \equiv \mathbf{Y}_C(\lambda f.\lambda xyz.\langle f, x, z\rangle)$,

where $\underline{S} \equiv \lambda nfx.f(nfx)$ and $\underline{0}$ are, respectively, the successor function and the numeral 0 of the Church numerals. Similarly, the nth Church numeral is denoted by $\underline{n}$. Moreover, the notations $\langle f, x\rangle$ and $\langle f, x, z\rangle$ denote Church pairing and, respectively, tripling functions.

As in [12], we have

Lemma 1. *$JM = J$ iff there exists an m such that $\mathbf{E}_{\mathcal{A}}\underline{m} = \mathbf{C}^*M$.*

Now, consider the following "points fixed" equations:

$$x\ \langle L', \underline{0}\rangle\ =\ \langle L', \underline{0}\rangle \tag{1}$$
$$x\ \langle L'', \underline{0}, \underline{1}\rangle\ =\ \langle L'', \underline{0}, \underline{1}\rangle \tag{2}$$
$$xQ\ =\ Q. \tag{3}$$

Now, if $M \equiv \lambda a.a(\mathbf{C}^*M_1)...(\mathbf{C}^*M_m)$ for some $M_i \in \mathcal{A}$ then

$$M\langle L', \underline{0}\rangle = L'M_1\underline{0}(\mathbf{C}^*M_2)...(\mathbf{C}^*M_m)$$
$$= \langle L', \underline{0}\rangle(\mathbf{C}^*M_2)...(\mathbf{C}^*M_m) = \cdots\ =\ \langle L', \underline{0}\rangle$$

and similarly

$$M\ \langle L'', \underline{0}, \underline{1}\rangle\ =\ \langle L'', \underline{0}, \underline{1}\rangle.$$

In addition,

$$\begin{aligned} MQ &= LP(\mathbf{C}^*M_1)...(\mathbf{C}^*M_m) \\ &= L(P(J(\mathbf{C}^*M_1)))(\mathbf{C}^*M_2)...(\mathbf{C}^*M_m) \\ &= L(PJ)(\mathbf{C}^*M_2)...(\mathbf{C}^*M_m) = LP(\mathbf{C}^*M_2)...(\mathbf{C}^*M_m) = \cdots = Q. \end{aligned}$$

Thus, all the members of the free monoid generated by the terms $\mathbf{C}^*M$ with $M \in \mathcal{A}$ satisfy Eqs. (1)–(3).

Proposition 4. *Suppose that N satisfies Eqs. (1)–(3), then N lies in the free monoid generated by the $\mathbf{C}^*M$ for M in $\mathcal{A}'$.*

Proof. Suppose that such an N is given. Since N satisfies Eq. (1), N has a head normal form. W.l.o.g. we may assume N is in head normal form. Since N satisfies Eq. (2), and terms $\langle L', \underline{0}\rangle$, $\langle L'', \underline{0}, \underline{1}\rangle$ have head variables with a different number of arguments, the head variable of N is the first one bound in its lambda prefix. Since $\langle L', \underline{0}\rangle$ has order 1, the lambda prefix of N has length 1 or 2. First, suppose that N has order 2; $N \equiv \lambda xy.xX_1...X_m$. Then, setting $Z_i \equiv [Q/x]X_i$

$NQ = \lambda y.QZ_1...Z_m = \lambda y.L(P(JZ_1))...(JZ_m)$. By an argument similar to the argument of [12] Theorem 3, this can only be the case if $Z_m = y$ and for $i < m$ we have $JZ_i = J$. Since Q contains an unprojectible free variable in F and G, it must be the case that each Z_i beta converts to a term without x, and for $i < m$ without y. In other words, x is head original and thus we may assume that

$$N = \lambda xy.xN_1...N_{m-1}y.$$

Hence, by Lemma 1, there exist $M_1, \ldots, M_{m-1}$ in $\mathcal{A}$ such that $N_i = \mathbf{C}^*M_i$ for $i = 1, \ldots, m-1$, and we have

$$N = \mathbf{B}\underline{1}(\mathbf{B}(\mathbf{C}^*M_1)(\cdots(\mathbf{B}(\mathbf{C}^*M_{m-2})(\mathbf{C}^*M_{m-1}))\cdots)).$$

The case for N of order 1 is similar with $\mathbf{I}$ replacing $\underline{1}$.

Remark. If the members of $\mathcal{A}'$ all have normal forms then the members of $\mathcal{A}''$ can be taken to be closed terms.

Acknowledgment

We thank the anonymous referee for her/his help to improve a previous version of the paper.

References

[1] H.P. Barendregt, Combinators & Lambda Calculus. www.cs.ru.nl/~henk/CT201014.pdf.
[2] H.P. Barendregt, The Lambda Calculus. Its Syntax and Semantics, North-Holland, 1984.
[3] H.P. Barendregt, W. Dekkers, R. Statman, Lambda Calculus with Types, Cambridge University Press, 2013.
[4] A. Church, A note on the entscheidungs problem, J. Symbolic Logic 1 (1936).
[5] J. Endrullis, D. Hendriks, J.W. Klop, Modular construction of fixed point combinators and clocked boehm trees, LICS (2010) 111–119.
[6] William Lawvere, Diagonal arguments and cartesian closed categories, Springer LNM 92 (1969) 134–145.
[7] Piergiorgio Odifreddi, S. Barry Cooper, Recursive Functions, in: Edward N. Zalta (Ed.) The Stanford Encyclopedia of Philos- ophy (Fall 2012 ed.), URL: http://plato.stanford.edu/archives/fall2012/entries/recursive-functions.
[8] Andrew Polonsky, Axiomatizing the quote, in: Proc. 20th Conference on Computer Science Logic - CSL'11, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 458–469.
[9] Yuan Qiaochu, Can the Lawvere fixed point theorem be used to prove the Brouwer fixed point theorem? MathOverflow Question 136478. Asked July 12th, 2013. http://mathoverflow.net/questions/136478.
[10] H. Rogers Jr., Theory of Recursive Functions and Effective Computability, MacGraw Hill, New York, 1967.
[11] T. Skolem, Bemerkungen zum komprehensionsaxiom, in: Zeitschrift fr Mathematische Logik Und Grundlagen Der Mathematik, Vol. 3, 1957, p. 117.
[12] R. Statman, Morphisms and partitions of V -sets, CSL'98.
[13] R. Statman, Some examples of non-existent combinators, Theoret. Comput. Sci. 121 (1993).
[14] R. Statman, On the representation of semigroups and other congruences in the lambda calculus, MFPS 2016, Electronic Notes in Theoretical Computer Science, The Thirty-second Conference on the Mathematical Foundations of Programming Semantics, MFPS XXXII, Pittsburgh, USA, May 23-26, 2016, pp. 16–23.
[15] A. Visser, in: Hindley, Seldin (Eds.), Numerations, Lambda Calculus, and Arithmetic, Curry Festschrift Academic Press, 1980, pp. 259–284.
[16] N.S. Yanofsky, A Universal Approach to Self-Referential Paradoxes, Incompleteness and Fixed Points, 2003. arXiv: math/0305282 [math.LO].