Available online at www.sciencedirect.com

ScienceDirect

Indagationes Mathematicae xx (xxxx) xxx–xxx

indagationes mathematicae

www.elsevier.com/locate/indag

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: benedetto.intrigila@gmail.com (B. Intrigila).

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

Please cite this article in press as: B. Intrigila, R. Statman, Fixed points in lambda calculus. An eccentric survey of problems and solutions, Indagationes Mathematicae (2017), 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:

Please cite this article in press as: B. Intrigila, R. Statman, Fixed points in lambda calculus. An eccentric survey of problems and solutions, Indagationes Mathematicae (2017), http://dx.doi.org/10.1016/j.indag.2017.06.003.