UNIVERSITÀ DEGLI STUDI DI “ROMA TRE”

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI
TESI DI LAUREA MAGISTRALE IN MATEMATICA

PUNTI-FISSI nel LAMBDA-CALCOLO

--- SINTESI ---

RELATORE

Lorenzo Tortora de Falco

REFERENTI

Hendrik Pieter Barendregt
Jan Willem Klop

CANDIDATO

Marco Buttafoco

ANNO ACCADEMICO 2009-2010
LUGLIO 2010

ABSTRACT

Fixed-Point are classical notions belonging to the heart of lambda-calculus and logic. We start with an explanation of the syntax of lambda-calculus, proving some principal properties and introducing the more general concept of reduction system. We then introduce combinators (lambda-terms without free variables) and the tree "famous" fixed-point combinator (defined by Curry, Turing and Klop). We prove that every combinator is "equivalent" to a lambda-term built using only combinators $\boldsymbol{S}$ *and* $\boldsymbol{K}$*, and we use fixed-point combinators to represent in the lambda-calculus every computable function.*
The well-known fact that if Y *is a fixed-point combinator then* $Y\delta$ *is again a fixed-point combinator, generates the Böhm sequence of fixed-point combinators.*
Using combinators $\boldsymbol{B}, \boldsymbol{I}$*, and* $\boldsymbol{S}$ *we build up two new sequences of FPCs, known as Scott and Klop sequences, and we present some generalization schemes to build infinitely many fixed-point combinators. In this way we find schemes and building blocks to construct new fixed-point combinators in a modular way. Having created a plethora of new fixed-point combinators, the task is to prove that they are indeed new, that is, we have to prove their* β*-inconvertibility. Known techniques using Böhm trees do not apply, because all fixed-point combinators have the same Böhm tree. One of the tools we use to distinguish fixed-point combinators is the notion of clocked Böhm tree, that convey information of the tempo in which the data in the Böhm trees are produced.*

INTRODUCTION

The theory of λ-*calculus* was introduced around 1930 by Alonzo Church [Chu41] as the kernel of an investigation in the foundation of mathematics and logic, in which the notion of "function" instead of "set" was taken as primitive. Subsequently, λ-calculus emerged as a consistent fragment of the original system, which became a key tool in the study of computability and, with the rise of computers, the formal basis of the functional programming paradigm. Today, λ-calculus plays an important role as a bridge between logic and computer science, which explains the general interest in this formalism among computer scientists.
A quarter of century after Barendregt's main book [Bar84], a wealth of interesting problems about models and theories of the (untyped) λ-calculus are still open.
After proving some important theorems described in [Bar84], the aim of this thesis is to create new fixed-point combinators from old ones. At the same time we will find a method for proving their β-inconvertibility.

OUTLINE

In *Chapter 1* we recall some known results concerning the (untyped and typed) λ-calculus and we introduce the more general concept of reduction system. In *Chapter 2* the combinatory logic is introduced: some fixed-point combinators (Curry, Turing, and Klop) are described, dealing with some important properties. After introducing a method to represent Church's integer and Booleans with lambda-terms, in *Chapter 3* we prove that all recursive functions are lambda-definable. *Chapter 4* is completely dedicated to Böhm-trees, the natural infinite generalization of normal forms in pure lambda-calculus. *Chapter 5* is devoted to the construction of new fixed-point combinators in modular way. In *Chapter 6* we show that all new fixed-point combinator are really new, that is we prove their not β-equivalence.

To keep this paper as self-contained as possible, first of all we summarize some definitions and results used below. With regard to the λ-calculus we follow the notation and terminology of [Bar84].

PUNTI-FISSI NEL LAMBDA-CALCOLO

The two primitive notions of the λ-calculus are application, the operation of applying a function to an argument, and lambda-abstraction, the process of forming a function from the "expression" defining it.

The set of *λ-terms* (notation Λ) over an infinite set of variables V using application and (function) abstraction is constructed inductively as follows: every variable $x \in V$ is a λ-term; if M and N are λ-terms, then so are MN and $\lambda x.M$ for each $x \in V$.

The lambda abstraction is a binder. An occurrence of a variable x in a λ-term is *bound* if it lies within the scope of a lambda-abstraction λx; otherwise it is *free*. We denote by $FV(M)$ the set of all free variables of M and we say that M is *closed* (or is a combinator) if $FV(M) = \emptyset$. We write $M[x \leftarrow N]$ or $M[x/N]$ for the term resulting from the substitution of N for all free occurrences of x in M subject to the usual proviso about renaming bound variables in M to avoid capture of free variables in N.

$$\boldsymbol{I} = \lambda x.x \qquad \boldsymbol{B} = \lambda xyz.x(yz) \qquad \boldsymbol{K} = \lambda xy.x \qquad \boldsymbol{S} = \lambda xyz.xz(yz).$$

The basic axioms of λ-calculus are the following:

(α) $\lambda x.M \rightarrow_\alpha \lambda y.M[x \leftarrow N]$, for any variable y that does not occur free in M.
(β) $(\lambda x.M)N \rightarrow_\beta M[x \leftarrow N]$.

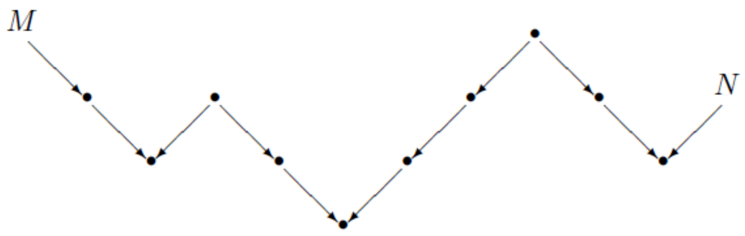
The rules for deriving equations from instances of (α) and (β) are the usual ones from equational calculus asserting that equality is a congruence for application and lambda-abstraction.
Extensional λ-calculus adds another axiom, which equates all the λ-terms having the same extensional behavior:

(η) $\lambda x.Mx \rightarrow_\eta M$, in which x does not occur free in M.

If two λ -terms are provably equal using the rule (α), we say that they are *α-equivalent* or *α-convertible* (and similarly for (β) and (η)). In this work we identify all α-equivalent λ-terms, thus every λ-term in fact represents a *class* of α-equivalent terms. We write $M \equiv N$ when M is α-equivalent to N.
Strictly speaking, $M \rightarrow N$ means that M reduces to N by exactly one reduction step, possibly applied to a subterm of M. Frequently, we are interested in whether M can be reduced to N by any number of steps. Write $M \twoheadrightarrow N$ if

$$M \rightarrow M_1 \rightarrow M_2 \rightarrow \cdots \rightarrow M_k \equiv N \quad (k \geq 0).$$

Informally, $M \simeq_\beta N$ if M can be transformed into M' by performing zero or more reductions and expansions (an expansion is the inverse of a reduction). A typical picture is the following:



If a term admits no reductions then it is in *normal form*. To normalize a term means to apply reductions until a normal form is reached. A term has a normal form if it can be reduced to a term in normal form.
A term is in *head normal form* (hnf, for short) if and only if it looks like this:

$$M \equiv \lambda x_1 \dots x_m . y M_1 \dots M_n$$

in which y is a variable ($m, n \geq 0$).

Notice that a term in normal form is also in hnf. Furthermore, if

$$\lambda x_1 \dots x_m . yM_1 \dots M_n \twoheadrightarrow_\beta N$$

then N must have the form

$$\lambda x_1 \dots x_m . yN_1 \dots N_n$$

in which $M_1 \twoheadrightarrow_\beta N_1, \dots, M_n \twoheadrightarrow_\beta N_n$. Thus, the hnf fixes the outer structure of any further reductions and the final normal form (if any!).

A λ-term is *defined* if and only if it can be reduced to head normal form; otherwise, it is *undefined*. A term is *solvable* if and only if there exist variables $x_1, \dots, x_k$ and terms $N_1, \dots, N_k,\ L_1, \dots L_l$ such that $(M[N_1/x_1, \dots, N_k/x_k])L_1 \dots L_l \simeq_\beta \boldsymbol{I}$. It can be proved that a term M is solvable if and only if it has a head normal form N if and only if there is a reduction sequence only composed of head steps such that $M \twoheadrightarrow N$ (one writes $M \twoheadrightarrow_h N$).

The *head reduction step* of a term M is the (finite or infinite) sequence of terms $M_0, M_1, \dots, M_n, \dots$ such that $M_0 = M$ and M_{n+1} is obtained from M_n by a β-reduction step of the head redex of M_n if such a redex exists; if not, M_n is in head normal form, and the sequence ends with M_n.

One of the fundamental relationships between the properties of reduction systems is the *Church-Rosser property* that states that reduction in λ-calculus is confluent; no two sequences of reductions, starting from one λ-term, can reach a distinct normal form. The normal form of a term is independent of the order in which reductions are performed.

For instance, $(\lambda x. ax)\big((\lambda y. by)c\big)$ has two different reduction sequences, both leading to the same normal form. The affected subterm is underlined at each step:

$$\underline{(\lambda x. ax)\big((\lambda y. by)c\big)} \to_\beta a\underline{\big((\lambda y. by)c\big)} \to_\beta a(bc)$$

$$(\lambda x. ax)\underline{\big((\lambda y. by)c\big)} \to_\beta \underline{(\lambda x. ax)(bc)} \to_\beta a(bc)$$

This property has several important consequences:

- if $M \simeq_\beta N$ and N is in normal form, then $M \twoheadrightarrow_\beta N$; if a term can transform into normal form through reductions and expansions, then the normal form can be reached by reductions alone;
- if $M \simeq_\beta N$ in which both terms are in normal form, then $M \equiv N$. If M and N are in normal form and are distinct, then $M \simeq_\beta N$.

Although different reduction sequences cannot yield different normal forms, they can yield completely different outcomes; one could terminate whereas the other runs forever.

Typically, if M has a normal form and admits an infinite reduction sequence, it contains a subterm L having no normal form, and L can be erased by a reduction.

For example, $\boldsymbol{\Omega} \equiv (\lambda x.xx)(\lambda x.xx) \to_\beta (\lambda x.xx)(\lambda x.xx) \to_\beta \dots$ β-reduces to itself and therefore no normal form exist.

The λ-calculus is expressive enough to encode Boolean values and natural numbers. More generally, all the data structures we may desire in a functional program. These encodings allow us to virtually model the whole of functional programming within the simple confines of λ-calculus. An encoding of the Booleans must define the terms true ($\mathbb{T}$) and false ($\mathbb{F}$), satisfying (for all terms M and N)

$$\mathbb{F}MN \to_\beta N$$

$$\mathbb{T}MN \to_\beta M$$

The following encoding is usually adopted:

$$\mathbb{F} \equiv \lambda xy.y$$

$$\mathbb{T} \equiv \lambda xy.x$$

We have $\mathbb{T} \not\simeq_\beta \mathbb{F}$ by the Church Rosser property, since true and false are distinct normal forms.

All the usual operations on truth values can be defined, so as conditional operator. Here conjunction, disjunction, and negation are described:

$$\boldsymbol{and} \equiv \lambda xy.xyx$$

$$\boldsymbol{or} \equiv \lambda xy.xxy$$

$$\boldsymbol{not} \equiv \lambda xyz.xzy$$

The following encoding of the natural numbers is the original one developed by Church. Define

$$\underline{0} = \lambda fx.x;$$

$$\underline{1} = \lambda fx.fx;$$

$$\underline{\dots} = \dots\dots\dots;$$

$$\underline{n} = \lambda fx.f^n x$$

in which $f^n x$ is defined by induction as follows:

- $f^1 x = fx$;
- $f^{n+1} x = f(f^n x)$.

Thus, for all $n \geq 0$, the Church numeral n acts as an iteration (n-times) of its original.

Using this encoding, successor, addition, and multiplication can be defined immediately:

$$\boldsymbol{succ}\,\underline{n} \equiv \lambda nfx.f(nfx)$$

$$\boldsymbol{sum}\,\underline{m}\,\underline{n} \equiv \lambda mnfx.mf(nfx)$$

$$\boldsymbol{prod}\,\underline{m}\,\underline{n} \equiv \lambda mnfx.m(nf)x$$

The operations defined so far are not sufficient to define all computable functions on the natural numbers. The secret is to use a *fixed-point combinator* (FPC), that is a term Y such that $YF \simeq_\beta F(YF)$, for all λ-terms F. Let us explain the terminology. A *fixed-point* of the function F is any X such that $FX \simeq_\beta X$; here, $X \equiv YF$. To code recursion, F represents the body of the recursive definition; the law $YF \simeq_\beta F(YF)$ permits F to be unfolded as many times as necessary. Moreover, an FPC Y is *reducing* (resp. k-*reducing*) if for any λ-term x we have $Yx \twoheadrightarrow_\beta x(Yx)$ (resp. $Yx \rightarrow^k x(Yx)$, in which k represent the number of the head reduction steps).

There exists infinitely-many FPCs. The most well known is due to Haskell B. Curry and is defined as follows:

$$Y_0 \equiv \lambda f.\omega_f\omega_f \equiv \lambda f.\big(\lambda x.f(xx)\big)\big(\lambda x.f(xx)\big).$$

Y_0 ia an FPC but not a reducing-FPC.

There are other FPCs, such as that of A. Turing:

$$Y_1 \equiv \eta\eta \equiv \big(\lambda xf.f(xxf)\big)\big(\lambda xy.f(xxf)\big)$$

We indeed have the reduction $Y_1F \twoheadrightarrow_\beta F(Y_1F)$.

From J.W. Klop's FPC,

$$\$ \equiv ££££££££££££££££££££££££££$$

in which $£ \equiv \lambda abcdefghijklmnopqstuvwxyzr.\big(r(thisisafixedpointcombinator)\big)$, we are able to build up a new class of FPCs [this resolves the Problem 6.8.14. in Bar84].

After showing some important properties of FPCs (such as one which proves that every FPC is solvable but not normalizable) we see that all operations of λ-calculus can be expressed in a reduced version of untyped λ-calculus, the so called SK-calculus. The importance of a fixed-point lies, however, in the fact that they allow us to solve equations. Therefore, FPCs are essential tools to represent all computable functions.

Since λ-calculus is one of the classical models of computation, along with Turing machines and general recursive functions, *Church's Thesis* states that the computable functions are precisely those that are λ-definable. After defining constant, projection, and minimization operators, we prove that λ-calculus has the same power as the partial recursive functions, i.e., that all partial recursive functions are λ-definable. Therefore, the following definition is fundamental.

Let φ be a (partial) function defined on $\mathbb{N}^p$, with values either in $\mathbb{N}$ or in $\{F, T\}$. We say that φ is *λ-definable* if there is a λ-term M_φ such that for all $(n_1, \dots, n_p) \in \mathbb{N}^p$ we have:

- if $\varphi(n_1, \dots, n_p)$ is undefined, then $\mathrm{M}_\varphi \underline{n}_1 \dots \underline{n}_p$ is not solvable;
- if $\varphi(n_1, \dots, n_p) = k$, then $\mathrm{M}_\varphi \underline{n}_1 \dots \underline{n}_p \twoheadrightarrow_\beta \underline{k}$;
- if $\varphi(n_1, \dots, n_p) = F$ (resp. $= T$), then $\mathrm{M}_\varphi \underline{n}_1 \dots \underline{n}_p \twoheadrightarrow_\beta \mathbb{F}$ (resp. $\twoheadrightarrow_\beta \mathbb{T}$).

We say that M_φ *represents* the function φ and φ is *representable* by the term M_φ.

The behavior of FPCs are characterized by *Böhm-trees* (BT, for short). We show how to progressively compute successive approximations of a term, in a potentially infinite partial structure called the Böhm-tree of a λ-term, which represents "the limit" of all β-reductions issued from a given term. It consists of layers of approximations, each approximation corresponding to an hnf. The formal definition of Böhm trees can be found with all its details in this thesis. Below, an informal definition of Böhm-tree is given. Note that this definition is not an inductive definition of $BT(M)$; the $N_1 \dots N_m$ in the tail of an hnf of a term may be more complicated then the term itself.

To each $M \in \Lambda$ a *Böhm-tree*, $BT(M)$, is associated as follows:

- if M is solvable then $M \twoheadrightarrow^h \lambda x_1 \dots x_n . y N_1 \dots N_m$ $(n, m \geq 0)$ and we define:

$$\begin{tikzcd} & BT(M) = \lambda x_1 \dots x_n . y \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\ BT(N_1) & \dots\dots\dots & BT(N_m) \end{tikzcd}$$

  i.e., level 0 of $BT(M)$ is known. By iteration we find all levels.
- otherwise, if M is unsolvable (equivalently, has no hnf) we have: $BT(M) = \perp$.

Notice that all unsolvable terms have the same BT, namely $\perp$. We prove in the thesis that all FPCs have the same BT with the same infinite extension $\lambda f . f(f(f \dots))$.

It is possible to show, by means of induction on the depth of the tree, that the given definition of BT is compatible with β-equivalence, i.e., that

$$M \simeq_\beta N \Rightarrow BT(M) = BT(N).$$

in which $M, N \in \Lambda$. The converse is not valid.

Owing to the equivalence of the BTs, we now propose to provide a method to inductively build up infinite FPCs. In particular, we can show how a new FPC (resp. reducing FPC) can be built up from a given FPC (resp. reducing FPC). To show this, we use the λ-term $\delta \equiv \lambda ab.b(ab)$. A first remark is that the term $Y\delta$ is an FPC whenever Y is. It follows that starting with Y_0 (Curry's FPC), we have an infinite sequence of FPCs

$$Y_0, Y_0\delta, Y_0\delta\delta, \dots, Y_0\delta^{\sim n}, \dots$$

in which $\delta^{\sim n}$ means $\delta\delta \dots \delta$ n-times, called the *Böhm sequence of* Y_0. The same sequence can also be defined as follows in which Y_1 is Turing's FPC:

$$\begin{array}{rcl} Y_0 & \equiv & \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \\ (Y_0\delta \simeq_\beta)\; Y_1 & \equiv & \eta\eta \\ Y_{n+2} & \equiv & Y_{n+1}\delta \text{ for } n \geq 1 \end{array}$$

Now the question is whether all these "derived FPCs" are really new, in other word, whether the sequence is free of duplicates [this resolves the Problem 6.8.9. in Bar84]. After defining, by induction on $n \in \mathbb{N}$, the following set of languages $\mathcal{L}_n \subseteq \Lambda$,

$$\mathcal{L}_0 \coloneqq \lambda f.f^k(\omega_f\omega_f) \qquad (k \geq 0)$$

$$\mathcal{L}_1 \coloneqq \eta\eta \mid \lambda f.f^k(\mathcal{L}_1 f) \qquad (k \geq 1)$$

$$\mathcal{L}_n \coloneqq \mathcal{L}_{n-1}\delta \mid \delta\mathcal{L}_n \mid \lambda b.b^k(\mathcal{L}_n b) \quad (n \geq 2, k \geq 1)$$

we show that the Böhm sequence contains no duplicates, that is:

for all $m, n \in \mathbb{N}$ such that $m \neq n$, we have $Y_n \not\simeq_\beta Y_m$.

For the sequence starting from an arbitrary FPC Y, it is actually an open problem whether the sequence of FPCs $Y, Y\delta, Y\delta\delta, \dots, Y\delta^{\sim n}, \dots$ is free of repetition. All we know, applying Intrigila's Theorem[1], is that no two consecutive FPCs in this sequence are β-convertible, i.e., for every FPC Y one has $Y \not\simeq_\beta Y\delta$.

Our next purpose is to build up a new sequence of FPCs starting from the combinators $\boldsymbol{B}, \boldsymbol{S}$, and $\boldsymbol{I}$ as defined above. To do this, we can consider, for any FPC Y, the following sequence:

$$\boldsymbol{B}Y, \boldsymbol{B}Y\boldsymbol{S}, \boldsymbol{B}Y\boldsymbol{SS}, \boldsymbol{B}Y\boldsymbol{SSS}, \dots, \boldsymbol{B}Y\boldsymbol{S}^{\sim n}, \dots$$

All these terms are not FPCs, but they are close to being FPCs since for the first two terms of the sequence, postfixing the combinatory $\boldsymbol{I}$ turns them into FPCs Y_0 and Y_1, and postfixing an $\boldsymbol{I}$ to all the terms in the sequence, yielding

[1] Problem n. 52 in the problem list of: N. Dershowitz, J.P. Jouannaud, J.W. Klop, *More problem in rewriting*, in Rewriting Techniques and Applications (C. Kirchner Ed.), Lectures Notes in Computer Science, volume 690, pages 468-487, Springer-Verlag, Berlin, 1993.

$$\boldsymbol{B}Y\boldsymbol{I}, \boldsymbol{B}Y\boldsymbol{SI}, \boldsymbol{B}Y\boldsymbol{SSI}, \boldsymbol{B}Y\boldsymbol{SSSI}, \dots, \boldsymbol{B}Y\boldsymbol{S}^{\sim n}\boldsymbol{I}, \dots$$

we obtain a sequence of terms having the same BT as any FCP. We refer to the previous sequence as the *Scott sequence of* Y.

In particular, we can consider the Scott sequence with $Y \equiv Y_0$ (Curry's FPC). In this way, we can define the λ-term $\mathcal{U}_n$ by induction on $n \in \mathbb{N}$ as follows:

$$\begin{aligned} \mathcal{U}_0 &\equiv \boldsymbol{B}Y_0\boldsymbol{I} \\ \mathcal{U}_1 &\equiv \boldsymbol{B}Y_0\boldsymbol{SI} \\ \mathcal{U}_n &\equiv \boldsymbol{B}Y_0\boldsymbol{S}^{\sim n}\boldsymbol{I} \end{aligned}$$

All these terms turn out to be FPCs, and in showing that $\boldsymbol{B}Y\boldsymbol{S}^{\sim n}\boldsymbol{I}$ are indeed FPCs, we find, as a bonus, a new FPC-generating vector, turning an old FPC into a new one: $Y(\boldsymbol{SS})\boldsymbol{S}^{\sim n}\boldsymbol{I}$. Let us call the derivation principle from Böhm, stating that postfixing a δ yields a new FPC: principle (δ). Now we have a second derivation principle, let us call it (σ), stating that postfixing to an FPC a vector of terms $(\boldsymbol{SS})$ $\boldsymbol{S}^{\sim n}\boldsymbol{I}$ yields a new FPC. We can arbitrarily apply derivation principles (δ) and (σ), and so obtain, starting from a given FPC, a whole branch of new FPCs.

In this way, we find some schemes and building blocks to construct new fixed-point combinators in a modular way. Through one of these schemes we can formulate the so called *Klop sequence*:

$$\boldsymbol{BBB}Y\boldsymbol{II}, \boldsymbol{BBB}YA\boldsymbol{II}, \boldsymbol{BBB}YAA\boldsymbol{II}, \dots, \boldsymbol{BBB}YA^{\sim n}\boldsymbol{II}, \dots$$

All terms of this sequence are FPCs and the sequence coincides (when $Y \equiv Y_0$) with the Böhm and Scott sequences for only the first two elements.

Since there are several vast spaces of FPCs and there are many ways to derive new FPCs, the last question is whether all these FPCs are indeed new. Therefore, we have to prove that they are not β-equivalent one another.

Since it is not possible to distinguish FPCs through their Böhm-tree (remember that any FPCs have the same BT), we use the *clocked Böhm-trees* (i.e., Böhm-trees in which we count the number of head reduction steps): we can discern a clock-like behavior of Böhm-trees, that enables us to discriminate the terms in question.

However, this refined discrimination method does not work for all lambda-terms; only for a class of simple terms, that still is fairly extensive.

Therefore, the following two definitions are essential.

For FPC Y, the *clock reduction* of Y consists of a sequence of head reduction steps (h-steps) and when no head steps are possible because the term is an hnf (i.e., in the form $\lambda x. xB$) there is a γ-step that removes the head context $\lambda x.[\quad]$ or $x[\quad]$ respectively:

$$\lambda x. xB \to_\gamma xB \text{ and } xB \to_\gamma B.$$

We than define simple-terms: a λ-term t is *simple* if if either t has no head normal form, or the head reduction to head normal form $t \twoheadrightarrow_h^k \lambda x_1 \dots x_n . yM_1 \dots M_m$ contracts only simple redexes (i.e., a linear or call-by-value redexes), and $M_1, \dots, M_m$ are simple terms. This notation leads to the following proposition:

if M and N are simple terms with different clocks, then $M \neq_\beta N$.

It is in this way that the non β-equivalence of the created fixed-point combinators can be shown.

Several examples can be found in the thesis. Below we write the “standard algorithm” for discriminate all FPCs:

(1) take a sequence of FPCs;
(2) reduce the sequence to a simple term;
(3) compute the clock reduction of the simple term;
(4) if all the simple terms have different clock, then FPCs are pairwise different.

We conclude by computing the algorithm for the sequence of FPC $\mathcal{U}_n \equiv \boldsymbol{B}Y_0\boldsymbol{S}^{\sim n}\boldsymbol{I}$ of the Scott sequence of Y_0, with $n \geq 2$.

We first reduce $\mathcal{U}_n$ to a simple term:

$$\mathcal{U}_n x \twoheadrightarrow Y_0(\boldsymbol{SS})\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x \to \omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x \twoheadrightarrow \underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x$$

where $\underline{\omega_{\boldsymbol{SS}}} = \lambda abc. bc(aabc)$. Then we compute the clock for $n = 2$, $n = 3$ and $n > 3$.

For $n = 2$ we have:

$$\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\,\boldsymbol{I}x \twoheadrightarrow_h^3 \boldsymbol{I}x\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{I}x\right) \to_h^1 x(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{I}x).$$

For $n = 3$ we have:

$$\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{SI}x \twoheadrightarrow_h^3 \boldsymbol{SI}\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{I}\right)x \twoheadrightarrow_h^4 x(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{SI}x).$$

For $n > 3$ we have:

$$\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x \twoheadrightarrow_h^3 \boldsymbol{SS}\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{SS}\right)\boldsymbol{S}^{\sim(n-4)}\boldsymbol{I}x \twoheadrightarrow_h^{3(n-4)}$$

$$\twoheadrightarrow_h^{3(n-4)} \boldsymbol{SS}\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{SSS}^{\sim(n-4)}\right)\boldsymbol{I}x \twoheadrightarrow_h^3$$

$$\twoheadrightarrow_h^3 \boldsymbol{SI}\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{SSS}^{\sim(n-2)}\boldsymbol{I}\right)x \twoheadrightarrow_h^4 x(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x).$$

For all cases, we find:

$$BT_{\text{🕘}}\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x\right) = [3n-2]\left(xBT_{\text{🕘}}\left(\underline{\omega_{\boldsymbol{SS}}\omega_{\boldsymbol{SS}}}\boldsymbol{S}^{\sim(n-2)}\boldsymbol{I}x\right)\right).$$

Therefore, given any two terms $\mathcal{U}_i$ and $\mathcal{U}_j$ of the Scott sequence of Y_0, if $i \neq j$ then $\mathcal{U}_i \neq_\beta \mathcal{U}_j$.

By losing some information (but keeping the essential), Figure 1 displays the clocked Böhm-trees of the Scott sequence of Y_0

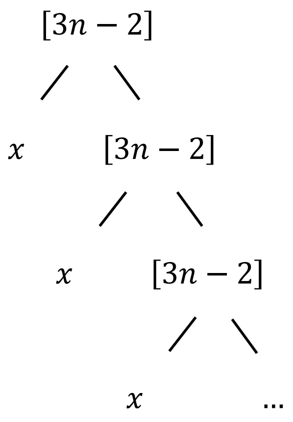


Figure 1 – clocked Böhm-tree of $\boldsymbol{\mathcal{U}_n}$

BIBLIOGRAFIA

[Abd76] ABDALI S.K.
[1976] *An abstraction algorithm for Combinatory Logic*, The Journal of Symbolic Logic, volume 41, number 1.

[Ada97] ADAMOWICZ Z., ZBIERSKI P.
[1997] *Logic of Mathematics*, Wiley Series in Pure and Applied Mathematics.

[Ama98] AMADIO R.M., CURIEN P.L.
[1998] *Domains and lambda-calculi*, Cambridge University Press.

[Ati69] ATIYAH M.F., MACDONALD I.G.
[1969] *Introduction to Commutative Algebra*, Westview Press.

[Aus03] AUSIELLO G., D'AMORE F., GAMBOSI G.
[2003] *Linguaggi, Modelli, Complessità*, Scienze e Tecnologie Informatiche, Franco Angeli Edizioni, Milano.

[Bar76] BARENDREGT H., BERGSTRA J., KLOP J.W., VOLKEN H.
[1976] *Some note on lambda reduction*, Chapter II in Degrees, reductions and representability in the lambda calculus, Technical Report Preprint number 22, University of Utrecht, Department of Mathematics.

[Bar84] BARENDREGT H.
[1984] *The Lambda Calculus: its syntax and semantics*, second, revisited edition, volume 103, North-Holland, Amsterdam.

[Bar92] BARENDREGT H., ABRAMSKY S., GABBAY D.M., MAIBAUM T.S.E.
[1992] *Lambda calculi with types*, Handbook of Logic in Computer Science, volume II, pages 117-309, Oxford University Press.

[Bar00a] BARENDREGT H., BARENDSEN E.
[2000] *Introduction to lambda calculus*, revised edition, University of Goteborg, Sweden.

[Bar00b] BARENDREGT H., GHILEZAN S.
[2000] *Lambda terms for natural deduction, sequent calculus and cut elimination*, Journal of Functional Programming, number 10, pages 121-134, Cambridge University Press.

[Bar09] BARENDREGT H., KLOP J.W.
[2009] *Application of Infinitary Lambda Calculus*, Information and Computation, volume 207, number 5, Pages 559-582.

[Bar10] BARENDREGT H., DEKKERS W., STATMAN R.
[2010] *Lambda calculus with types*, Radboud University, Nijmegen, the Netherlands (to be published).

[Ber80] BERGSTRA J., KLOP. J.W.
[1980] *Invertible Terms in the Lambda-Calculus*, Theoretical Computer Science, volume 11, number 1, pages 19-37.

[Bez03] BEZEM M., KLOP J.W., DE VRIJER R.
[2003] *Term rewriting systems* ("Terese"), Cambridge University Press.

[Böh68] BÖHM C.
[1968] *Alcune proprietà delle forme β-η-normali nel λ-K-calcolo*, Pubblicazioni dell'IAC, numero 696, pagine 1-19.

[Böh75] BÖHM C.
[1975] *λ-calculus and computer science theory*, SLNCS 37, Springer-Verlag, Berlin.

[Bro76] BROWDER F.E..
[1976] *Mathematical developments arising from Hilbert Problems,* Proceedings of symposia in pure mathematics, American Mathematical Society, volume XXVIII.

[Chu36a] CHURCH A., ROSSER J.B.
[1936] *Some Properties of Conversion*, Transaction of the American Mathematical Society, volume 39, number 3, pages 474-482.

[Chu36b] CHURCH A.
[1936] *An unsolvable Problem of Elementary Number Theory*, American Journal of Mathematics, volume 58, number 2, pages 345-363.

[Chu40] CHURCH A.
[1940] *A formulation of the simple theory of types*, The Journal of Symbolic Logic, volume 5, number 2: pages 56-68.

[Chu41] CHURCH A.
[1941] *The Calculi of Lambda-Conversion*, Princeton University Press, Annals of Mathematics Studies, number 6.

[Cor01] CORI R., LASCAR D.
[2001] *Mathematical Logic: A Course with Exercise, Part I-II*, traslated by D.H. Pellefier, Oxford University Press.

[Cor09] CORMEN T.H., LEISERSON C.E., RIVEST R.L., STEIN C.
[2009] *Introduction to algorithm*, McGraw-Hill, Third edition.

[Cur98a] CURIEN P.L.
[1998] *Abstract Böhm trees*, Mathematical Structures in Computer Science, number 8, pages 559-591.

[Cur98b] CURIEN P.L., HERBELIN H.
[1998] *Computing with abstract Böhm trees*, Third Fuji International Symposium on Functional and Logic Programming, Kyoto, Eds Masahiko Sato & Yoshihito Toyama, World Scientific (Singapore), pages 20-39.

[Cur34] CURRY H.B.
[1934] *Functionality in Combinatory Logic*, Proceedings of the National Academy of Sciences USA, volume 20, pages 584-590.

[Cur58] CURRY H.B., FEYS R.
[1958] *Combinatory Logic*, Studies in Logic and the Foundations of Mathematics, volume I, North-Holland, Amsterdam.

[Cur72] CURRY H.B.
[1972] *Combinatory Logic*, Studies in Logic and the Foundations of Mathematics, volume II, North-Holland, Amsterdam, Elsevier.

[Dav99] DAVIS P.J., HERSH R..
[1999] *The Mathematical Experience*, Meriner Books.

[Dav03] DAVIS M.
[2003] *Il calcolatore universale: da Leibnitz a Turing*, traduzione di G. Rigamonti, Adelphi, Milano.

[Dez98] DEZANI-CIANCAGLINI M., INTRIGILA B., VENTURINI-ZILLI M.
[1998] *Böhm's theorem for Böhm trees*, Theoretical Computer Science (Prato), pages 1-23. World Science Publishing, River Edge, New Jersey.

[Dub04] DUBBEY J.M.
[2004] *The Mathematical Work of Charles Babbage*, Cambridge University Press.

[End10] ENDRULLIS J., GRABMAYER C., HENDRIKS D., ISIHARA A., KLOP J.W.
[2010] *Productivity of Stream Definition*, Graphical Lambda Calculator, in http://joerg.endrullis.de/lambdaCalculator/index.html and http://fspc282.few.vu.nl/productivity/lambdaCalculator.html.

[Gan36] GANDY R.
[1936] *The Confluence of Ideas*, A Half-Century Survive on The Universal Turing Machines, Oxford University Press, pages 55-111.

[Gau02] GAUTHIER Y.
[2002] *Internal Logic, Foundation of Mathematics from Kronecker to Hilbert*, Volume 310, Kluwer Academic Publisher.

[Gel09] GELENBE E., KAHANE J.P.
[2009] *Böhm's Theorem*, Chapter 1 in Fundamental Concepts in Computer Science, Advances in Computer Science and Engineering: Texts, volume 3, Imperial College Press.

[Ger96] GERSTEIN L.J.
[1996] *Introductions to Mathematical Structures and Proofs*, pages 48-49, Springer-Verlag, New York.

[Göd31] GÖDEL K.
[1931] *On formally undecidable propositions of principia mathematica and related systems*, Dover edition, English translation of Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, published in the Monatshefte für Mathematik und Physik, Volume 38, pages 173,198.

[Gol05] GOLDBERG M.
[2005] *On the Recursive Enumerability of Fixed-Point Combinator*, BRICS, Department of Computer Science, University of Haarhus.

[Hei67] VAN HEIJENOORT J.
[1967] *From Frege to Gödel, Letter to Frege (1902)*, A Source Book in Mathematical Logic, pages 124-125.

[Her97] HERSH R..
[1997] *What is Mathematics, really?*, Oxford University Press.

[Hin72] HINDLEY J.R., LERCHER B., SELDING S.P.
[1972] *Introduction to Combinatory Logic*, Volume II, North-Holland, Amsterdam.

[Hin86] HINDLEY J.R., SELDING S.P.
[1986] *Introduction to Combinators and λ-Calculus*, Cambridge University Press, New York, USA.

[Hue75] HUET G.
[1975] *Unification in Typed Lambda Calculus*, in λ-calculus and Computer Science Theory. Proceedings of the Symposium Held in Rome, March 25-27, pages 192-212, Springer-Verlog.

[Hue93] HUET G.
[1993] *An analysis of Böhm's theorem,* Theoretical Computer Science, number 121, pages 145-167.

[Hyl76] HYLAND M.
[1976] *A syntactic characterization of the equality in some models for the lambda calculus*, Journal of the London Mathematical Society, volume 12, number 3, pages 361-370.

[Int97] INTRIGILLA B.
[1997] *Non-existent Statman's Double Fixed Point Combinator Does Not Exist, Indeed*, Information and Computation, number 137, pages 35-40

[Jun04] JUNG A.
[2004] *A short introduction to the Lambda Calculus*, University of Birmingham.

[Kas00] KASHIMA R.
[2000] *A Proof of the Standardization Theorem in λ-calculus*, Department of Mathematical and Computer Science, Tokio Institute of Tecnology, Ookayama, Meguro, Tokio, 152-8552, Japan.

[Ken95a] KENNAWAY J.R., KLOP J.W., SLEEP M.R., DE VRIES F.J.
[1995] *Infinitary lambda calculus and Böhm models*, Proc. Conference on rewriting Techniques and Application, SLNCS 914, Springer, pages 257-270.

[Ken95b] KENNAWAY J.R., KLOP J.W., SLEEP M.R., DE VRIES F.J.
[1995] *Transfinite Reductions in Orthogonal Term Rewriting System*, Information and Computation, volume 119, number 1, pages 18-38.

[Ken96] KENNAWAY J.R.
[1996] *Transfinite Rewriting*, International School on Type Theory and Term Rewriting, Glasgow.

[Ken97] KENNAWAY J.R., KLOP J.W., SLEEP M.R., DE VRIES F.J.
[1997] *Infinitary lambda calculus*, Theoretical Computer Science 175, volume 1, pages 93-125, Non-standard logics and logical aspects of computer science (Kanazawa, 1994).

[Ket05] KETEMA J., SIMONSEN J.G.
[1995] *Infinitary combinatory reduction system*, in J. Giesl (ed.), Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA 2005), number 3467 in Lecture Notes in Computer Science, Spinger-Verlag, pages 438-452.

[Kle34] KLEENE S.C., ROSSER J.B.
[1934] *The inconsistence of certain formal logics*, Annales of Mathematics, volume 36 number 3, pages 630-637.

[Kle36] KLEENE S.C.
[1936] *Lambda-Definability and Recursiveness*, Duke Mathematical Journal, volume 2 pages 340-353.

[Klo00] KLOP J.W., VAN OOSTROM V., DE VRIJER R.
[2000] *A geometric proof of confluence by decreasing diagram*, Journal of Logic and Computation, volume 10, number 3, pages 437-460.

[Klo05] KLOP J.W., DE VRIJER R.
[2005] *Infinitary normalization*, in S.N. Artemov, H. Barringer, A.S. d'Avila Garcez, L.C. Lamb and J. Woods (eds.), We will show them: Essay in honor of Dov Gabbay, volume 2, College Publications, pages 169-192.

[Klo07] KLOP J.W.
[2007] *New fixed point combinatory from old*, Chapter 16 in Reflection on Type Theory, λ-Calculus and the Mind, Barensen E., Capretta V. Geuvers H., Niqui M. editors.

[Klo10] KLOP J.W., ENDRULLIS J., HENDRIKS D.
[2010] *Personal communication during the meeting in The Netherlands.*

[Kri93] KRIVINE J.R., CORI R.
[1993] *Lambda-calculus: types and models* (English version), Ellis Horwood Series in Computer and their Application, Masson and Ellis Horwood.

[Lel33] LELAND LOCKE L.
[1933] *The Contribution of Leibniz to the art of Mechanical Calculation*, in Scripta Mathematica I, edited by Ginsburg J., pages 315-321.

[Lon83] LONGO G.
[1983] *Set theoretical models of λ-calculus, isomorphisms*, Annals of Pure and Applied Logic, volume 24, pages 153-188.

[Mit79] MITSCHKE G.
[1979] *The standardization theorem of λ-calculus*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, volume 25, number 1, pages 29-31.

[New80] NEWMAN M.H.A.
[1980] *On theories with a combinatorial definition of 'equivalence'*, Annales of Mathematics, volume 43, number 2, pages 223-243.

[Odd89] ODDIFREDDI P.
[1989] *Classical Recursion Theory: The Theory of Function and Sets of Natural Numbers*, volume 125, North-Holland, Amsterdam

[Ped08] PEDICINI M.
[2008] *Dispense ed esercizi del corso di IN2 – Modelli di calcolo*, Dipartimento di Matematica, Università degli Studi di "Roma Tre".

[Poi97] POINCARÈ J.-H.
[1936] *Scienza e metodo*, Einaudi, Torino.

[Sch24] SCHÖNFINKEL M.
[1924] *Űber die Bausteine der mathematischen Logik*, Mathematische Annalen 92, pages 305-316. Translated by Stefan Bauer-Mengelberg as "On the building blocks of mathematical logic" in Jean van Heijenoort [1967]. A Source Book in Mathematical Logic, 1879-1931 (pages 355-366), Harvard University Press.

[Sco75] SCOTT D.
[1975] *Some Philosophical Issue concerning Theories of Combinators*, in λ-calculus and Computer Science Theory. Proceedings of the Symposium Held in Rome, March 25-27, pages 346-366, Springer-Verlog.

[Sel01] SELINGER P.
[2001] *Lecture notes on the Lambda Calculus*, Department of Mathematics and Statistics, University of Ottawa.

[She03] SHEN A., VERESHCHAGIN N.K.
[2003] *Computable Functions*, Student Mathematical Library, Volume 19, American Mathematical Society.

[Sta93] STATMAN R.
[1993] *Some Example of Non-Existent Combinator*, Theoretical Computer Science, volume 121, Issue 1-2, pages 441-448.

[Sør06] SØRENSEN M.H., URZYCZYN P.
[2006] *Lecture on the Curry-Howard Isomorphism*, Studies in logic and the foundations of mathematics, volume 149, Elsevier.

[Tor08] TORTORA DE FALCO L.
[2008] *Dispense ed esercizi del corso MC4 – Matematiche Complementari 4, Logica classica del primo ordine*, Dipartimento di Matematica, Università degli Studi di "Roma Tre".

[Tur36] TURING A.
[1936] *On computable numbers, with an applicative to the Entscheidungproblem*, Proceedings of the London Mathematical Society, volume 42, issue 2, pages 230-265.

[Wad71] WADSWORTH C.P.
[1971] *Semantic and pragmatics of the lambda-calculus*, D. Phil Thesis, Oxford University.

[Wad76] WADSWORTH C.P.
[1976] *The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus*, Siam Journal of Computing, volumen 5, issue 3, pages 488-521.