λ-calculus is not an exotic mathematical subject

Davide Barbarossa*

November 4, 2022

Abstract

If your background is from general mathematical studies, I believe it is possible that your first impressions reading about λ-calculus are somehow confusing[1]. At least, such was my experience. Some concepts *seem* new; for example, it could seem that λ-calculus has an weird construction as a formal[2] object; or that the notion of denotational semantics is proper to this discipline and there is no relation with other familiar concepts in more mainstream parts of mathematics. In my opinion, this is due to historical reasons: λ-calculus started as a branch of mathematical logic – a field with its own lexicon and problems – and it was then developed at the periphery of mainstream mathematics[3]. In this discursive note I will try to show how the main areas of λ-calculus, starting from its very definition as a formal object, to denotational semantics and λ-theories, are actually nothing exotic and a parallel can be made with one of the most pervasive notions in mathematics, that of polynomials. My point is that if things were presented in the same manner one does for polynomials in usual mathematics courses, and not in a manner coming from the tradition of mathematical logic, many more mathematicians (including the theoretical computer scientists) would feel it as a familiar subject.

This note is organised as follows: I first consider the definition of λ-calculus by making a parallel with the definition of polynomials; I also mention the main reason why it is an interesting mathematical subject; then I consider the notions of denotational semantics and λ-theories, again in parallel with what one does with polynomials even if employing a different lexicon.

1 The definition...

1.1 ...of polynomials

Consider the construction of the polynomial algebra over a ring (or a field) $\mathbb{K}$. There are many equivalent ways to construct it, a possible one is the following:

*Thanks to Tito for some suggestions on the presentation!

[1] I guess this is also the case for most of other subject in related areas, such as mathematical logic and particularly proof-theory.

[2] In the sense usually employed by mathematicians. Logicians and computer scientists would say "syntactical".

[3] For instance, nowadays it is mainly done in computer science departments.

- First, you fix a finite set of distinct formal symbols $x_1, \dots, x_n$, called indeterminates (or variables) and define by induction a set $\hat{\mathbb{K}}[x_1, \dots, x_n]$ of words[4] (on a non specified yet clear from what follows alphabet), called *polynomial expressions*, by saying that:
  - the word x_i belongs to $\hat{\mathbb{K}}[x_1, \dots, x_n]$ for all $i = 1, \dots, n$;
  - if $p, q \in \hat{\mathbb{K}}[x_1, \dots, x_n]$ and $a \in \mathbb{K}$, then the words ap, $p + q$ and pq belong to $\hat{\mathbb{K}}[x_1, \dots, x_n]$.

  Remark that the symbol $+$, and the fact of making the "product" of an element of $\mathbb{K}$ and of $\hat{\mathbb{K}}[x_1, \dots, x_n]$, or of two elements of $\hat{\mathbb{K}}[x_1, \dots, x_n]$, do not have any meaning, they are just formal symbols making a word.
- Then you define the quotient set $\mathbb{K}[x_1, \dots, x_n] := \hat{\mathbb{K}}[x_1, \dots, x_n]/_\sim$, for a certain equivalence $\sim$ on $\hat{\mathbb{K}}[x_1, \dots, x_n]$ (whose definition I do not write here) consisting in the natural equations making it an algebra over $\mathbb{K}$, called the *polynomial algebra over* $\mathbb{K}$ *on* $x_1, \dots, x_n$. Its elements are called *polynomials*.

This is a somehow unusual construction. A common one is simply to say that the polynomial algebra $\mathbb{K}[x_1, \dots, x_n]$ is the free $\mathbb{K}$-algebra on $\{x_1, \dots, x_n\}$. There are several other equivalent possible definitions. I chose to take the unusual one just in order to push even further the analogy with the inductive definition of λ-calculus, but my point remains valid with whatever definition you prefer.

Choosing $n = 1$ just for simplicity, you can easily see that any polynomial $p \in \mathbb{K}[x]$ can be written in the canonical shape $p = \sum_{i=1}^{n} a_i x^i$, for some unique $a_1, \dots, a_n \in \mathbb{K}$ (meaning that, as an equivalence class, p admits a polynomial expression of that shape).

1.2 ...of λ-calculus

In perfect analogy with the construction of polynomials, λ-calculus is defined as follows[5]:

- First, you fix a countably infinite set of distinct formal symbols $x_1, x_2, x_3, \dots$, called variables and define by induction a set $\hat{\Lambda}$ of words[6] (on a non specified yet clear from what follows alphabet), called λ-*expressions*, by saying that:
  - the word x_i belongs to $\hat{\Lambda}$ for all $i \in \mathbb{N}$;
  - if $p, q \in \hat{\Lambda}$ and x is a variable, then the words $\lambda x.p$ and pq belong to $\hat{\Lambda}$.

  Analogously to the case of $\hat{\mathbb{K}}[x_1, \dots, x_n]$, remark that the symbols λ and $.$ in $\lambda x.f$, as well as making the "product" of two elements of $\hat{\Lambda}$, do not have any meaning, they are just formal symbols making a word.
- Then you define the quotient set $\Lambda := \hat{\Lambda}/_\sim$, for a certain equivalence $\sim$

[4] In order for the definition to be rigorously correct you should handle parentheses. I will not do it because it is clear what one means. Actually, it would be more elegant to define them as trees, which would also need no parentheses. I do not define them in that way just for the sake of brevity.

[5] Unlike for polynomials, for λ-calculus this is the construction given the vast majority of the time. Another definition would be as the internal language of particular kinds of Cartesian closed categories.

[6] See footnote 4.

on $\hat{\Lambda}$ called α-equivalence (whose definition I do not write here[7]). Its elements are called λ-*terms*.

From the definition of the α-equivalence, it is easily seen that any λ-term $p \in \Lambda$ can be written in a way s.t. all the λ-abstracted variables (i.e. the variables which immediately follow a λ) are pairwise different (in the sense that, as an equivalence class, p admits a λ-expression with that property). Furthermore, it can be written in the canonical shape $p = \lambda x_{i_1} \dots \lambda x_{i_m}.(((hq_1)\dots)q_n)$ (meaning that, as an equivalence class, p admits a λ-expression of that shape), for some unique $m, n \in \mathbb{N}$, $x_{i_1}, \dots, x_{i_m}$ variables and $h, q_1, \dots, q_n \in \Lambda$, where h is either a variable (equal or not to one of the x_{i_j}'s) or it is of shape $h = (\lambda y.h_1)h_2$, for some $h_1, h_2 \in \Lambda$ and y a variable different from all the x_{i_j}'s.

Notation 1.1. *In computer science and logic, one likes to write inductive definitions in a compact way. For instance, the of sets of words* $\hat{\mathbb{K}}[x_1, \dots, x_n]$ *and* $\hat{\Lambda}$ *would be respectively written as follows:*

$$p ::= x \ \mid \ ap \ \mid \ p + p \ \mid \ pp \qquad \text{for } x \in \{x_1, \dots, x_n\}$$

$$p ::= x \ \mid \ \lambda x.p \ \mid \ pp \qquad \text{for } x \in \{x_1, x_2, x_3, \dots\}.$$

2 Why one is interested...

2.1 ...in polynomials

Of course polynomials are interesting for a thousand reasons. Let us say that the main feature is that, under the operations $(a, p) \mapsto ap$, $(p, q) \mapsto pq$ and $(p, q) \mapsto p + q$, the set $\mathbb{K}[x_1, \dots, x_n]$ is a commutative algebra over $\mathbb{K}$.

2.2 ...in λ-calculus

For λ-calculus, the main feature that usually motivates its study is that, under a certain relation $\to_\beta \subseteq \Lambda \times \Lambda$ (called the β-reduction and whose definition I do not write here), the rewriting system $(\Lambda, \to_\beta)$ is a confluent and Turing-complete programming language. Furthermore, and this is the crucial point, it can be treated as a mathematical object with a rich and non-trivial mathematical theory (this is not quite the case for the other programming languages, even

[7] Let me still explain it without giving the rigorous definition. Actually, you already know it from high-school, when you learnt the "equalities" $\sum_i a_i = \sum_j a_j$, or $\int f(x)dx = \int f(t)dt$. They say that the name of the index variable of a summation, or the name of the integration variable, do not matter and you can use the name that you prefer (by paying attention to not use a name that for some reason is already used in the the terms of the summation or in the function integrated): all it matters is *which* index inside the terms of the summation, or variable inside the function integrated, they refer to. In computer science we say which index/variable they "capture", or also that the sum symbol $\sum_i \_$ and integral symbol $\int \_\, dx$ *bind* the index i/variable x. In the same way, α-equivalence says that $\lambda x.p\{x\} = \lambda y.p\{y\}$, where by $p\{x\}$ I mean that the λ-expression p depends on a variable x (the precise notion is called the fact that x is "free in p"). Again, it says that the $\lambda x.\_$ symbol binds the variable x, i.e. you can change its name as you like (by paying attention to not use one that for some reason is already used in p). In the case of sums and integrals those are not real equalities because it is just a matter of notation. Since λ-calculus is a syntactical object, α-equivalence has to be defined as a real equivalence on the set of λ-expressions.

the theoretical ones[8]). It is also a privileged setting[9], or it extensible to such setting, where one can study some programming and other computer science notions in a theoretical way. This is why it is considered as the "core" of all functional programming languages (real life ones included).

3 Denotational semantics...

3.1 ...for polynomials

3.1.1 As interpretation maps

Polynomials are formal expressions. However, one may want to see them as functions. This is easily done, since every polynomial $p \in \mathbb{K}[x_1, \ldots, x_n]$ defines a function $[\![p]\!] : \mathbb{K}^n \to \mathbb{K}$ in the natural way (first make a formal substitution, then evaluate the formal expression in $\mathbb{K}$), called the polynomial function associated with p (in logic or computer science we would say the *interpretation* of p). Of course there are many other ways to associate a function with a polynomial. For instance the trivial one associating the 0 constant function to every polynomial, or dumb ones, like the one that associate the 0 constant function with every polynomial of $\mathbb{K}[x]$ except for $x \in \mathbb{K}[x]$, which is associated with the 42 constant function. The trivial one is really not interesting, and the reason why the dumb one is a dumb one, is because $[\![.]\!]$ is not an algebra-homomorphism between $\mathbb{K}[x]$ and $\mathbb{K}^{\mathbb{K}}$. On the contrary, of course, interpreting polynomials as polynomial functions is indeed an *algebra-homomorphism*, and this is the natural requirement to ask to our interpretation map. In this way we answer to the question: *Can we see a polynomial as a function, in an interesting way?* In λ-calculus, we will see that there is instead an additional natural requirement to ask.

3.1.2 As quotient rings by ideals

Given the polynomial function interpretation map $[\![.]\!]$, this induces as usual an equivalence $=_{[\![.]\!]}$ on $\mathbb{K}[x_1, \ldots, x_n]$ by taking the fibres of $[\![.]\!]$. For $\mathbb{K}$ a field, it is well known that $[\![.]\!]$ is an isomorphism between $\mathbb{K}[x_1, \ldots, x_n]$ and $\mathbb{K}^{\mathbb{K}^n}$ iff $\mathbb{K}$ is infinite. For example, by Fermat's little theorem we have $x^k = x$ for all $x \in \mathbb{F}_k$ (the field with a prime number k of elements), while $x^k \neq x$ in $\mathbb{F}_k[x]$. However, there is another way of making interesting quotient polynomial rings, namely by quotienting w.r.t. polynomial ideals. For example, even if $\mathbb{R}$ is infinite, in the quotient ring $\mathbb{R}[x]/(x^2 - 1)$ we have $x^3 = x$. The canonical map to the quotient is thus an interesting interpretation map. Note that this interpretation does not directly answer to the question of seeing a polynomial as a function (for example, interpreting $x^3 \in \mathbb{R}[x]$ as the identity function on $\mathbb{R}$ would not give rise to an homomorphism). As you will see, also this way of identifying polynomials by *quotienting w.r.t. "compatible" equivalences* (here the one induced by the

[8]For the theoretical ones, this not exactly true. For instance, Martin-Löf Type-Theories are a class of programming languages with a rich mathematical theory, particularly regarding their relations with groupoids, Homotopy Type Theory and in general higher-category theory. However they are typically not Turing-complete, and by the way contain λ-calculus as a core computational fragment.

[9]This is particularly true for the *functional* paradigm. A similar role for the *imperative* one is played by Turing-machines.

polynomial ideal), has an analogue in λ-calculus. Compatible means that the quotient by such equivalence maintains the same algebraic structure; in the case of the $\mathbb{K}$-algebra $\mathbb{K}[x_1, \ldots, x_n]$, its quotients w.r.t. polynomial ideals are still $\mathbb{K}$-algebras.

3.2 ...for λ-terms

3.2.1 As interpretation maps

In λ-calculus, there is *a priori* no immediate candidate codomain for an interpretation map[10]. On Λ we have two operations, called respectively "application" and "λ-abstraction": $(p, q) \in \Lambda \times \Lambda \to pq \in \Lambda$ and $(x, p) \in \{x_1, x_2, x_3, \ldots\} \times \Lambda \subseteq \Lambda \times \Lambda \to \lambda x.p \in \Lambda$. Following the case of polynomials, we would like to define an interpretation $[\![.]\!]$ which is a homomorphism w.r.t. to those two operations. Since those two operations do not fall under some usual algebraic terminology, instead of saying that we want $[\![.]\!]$ to be a homomorphism, one says that we want it to be *compositional*. The intuition behind this terminology is that one is able to compute $[\![p]\!]$ by computing it on the "subterms" of p (roughly its sub-routines, if you think of p as a program). As for polynomials, we want to answer (if possible) the question: *Can we see a λ-term as a function, in an interesting way?* Here interesting means, as for polynomials, compositional. In λ-calculus the situation is more complex than for polynomials. It turns out that in order to answer the question one cannot use the usual set-theoretic notion of function; we must move to the abstract notion of function as morphism in a category.

As we already mentioned, there is another natural requirement to ask: in λ-calculus, we do not only have the above mentioned two operations; we also have the fundamental rewriting relation $\to_\beta$. It is therefore natural to ask whether we can define an interpretation map $[\![.]\!]$ from Λ to some set in such a way that it *passes to the quotient* $\Lambda/_{=_\beta}$, where $=_\beta$ is the equivalence generated by $\to_\beta$. This question is interesting because it can be read from the following two points of view:

1. The rewriting relation $\to_\beta$ endows Λ with a dynamics[11]. Physics teaches us that in a dynamical system it is always important to study the invariants of the dynamic. The idea is then to associate a λ-term with a morphism between two objects (thought of as "spaces") of a suitable category, in an invariant way w.r.t. $\to_\beta$, i.e. in such a way that if $p \to_\beta q$ then $[\![p]\!] = [\![q]\!]$. In that setting one can positively answer to the question: Can I see a λ-term as a function, in a compositional and $\to_\beta$-invariant way? The short answer is that in the untyped case, a λ-term can be seen as a morphism in any Cartesian closed category $\mathcal{C}$ with a reflexive object U (note that the category **Set** of sets and functions is Carte-

[10] This may not be completely true, depending on your familiarity with the subject. For instance, looking at the simply-typed λ-calculus one is immediately tempted to interpret simply-typed λ-terms as functions between sets in a natural way. In that case, it works. Also in the case of untyped λ-calculus – the one presented here – one could be tempted to do the same. But it is well known that in this case it does not work.

[11] $(\Lambda, \to_\beta)$ is a non-deterministic discrete dynamical system. In can be turned into a deterministic one by restricting $\to_\beta$ to a function $\Lambda \to \Lambda$. Such functions are known as "reduction strategies", and can be seen as functions $S : \Lambda \to \Lambda$ s.t. $p \to_\beta S(p)$. Given such S, the function $(n, p) \in \mathbb{N} \times \Lambda \to S^n(p) \in \Lambda$ is by construction a deterministic discrete dynamical system.

sian closed but it does not admit non-trivial reflexive objects), via an interpretation map $[\![.]\!] : \Lambda \to \mathrm{Hom}_{\mathcal{C}}(U, U)$. In the simply-typed case a simply-typed λ-term can be seen as a morphism in any Cartesian closed category, via an interpretation map $[\![.]\!] : \mathrm{ST}\Lambda \to \bigcup_{X,Y} \mathrm{Hom}_{\mathcal{C}}(X, Y)$ s.t. $[\![\Gamma \vdash p : A]\!] \in \mathrm{Hom}_{\mathcal{C}}([\![\Gamma]\!], [\![A]\!])$.

2. In the context of rewriting systems it is often interesting to consider the elements (p, q) of the equivalence induced by a rewriting rule as axiomatic equations $p = q$. One then wants to find the structures that satisfy this axioms, which logicians call the *models* of the axioms. For instance, if you suppose to have symbols $1, 1^{-1}, g_1, g_1^{-1}, g_2, g_2^{-1}, \ldots$, and you take the rewriting rules $gg^{-1} \to 1, g^{-1}g \to 1,\ g1 \to g,\ 1g \to g$ (for g any of your symbols), then the induced equivalence is of course describing the notion of group, in the sense that the axiomatic equations are those of the definition of groups. So their models are precisely the groups. In the case of λ-calculus, thus, we can see the image of the interpretation map $[\![.]\!]$ as a structure that validates all axioms of shape $p =_\beta q$. It turns out that one can give an exact algebraic characterisation of those structures, in terms of what are known as *combinatorial algebras*.

3.2.2 As quotient combinatory-algebras by λ-theories

Given an interpretation map $[\![.]\!]$ (with whatever codomain may it be), this induces as usual an equivalence $=_{[\![.]\!]}$ on Λ by taking the fibers of $[\![.]\!]$. Since $[\![.]\!]$ is compositional, then $=_{[\![.]\!]}$ is closed w.r.t. λ-abstraction and application. Since $[\![.]\!]$ is invariant for $=_\beta$, then $=_{[\![.]\!]} \supseteq =_\beta$. When an equivalence relation on Λ satisfies this two conditions (closure and contains $=_\beta$), it is called a *λ-theory*. So an interpretation map induces a λ-theory. The converse is also true: given a λ-theory $\sim$, the canonical map to the quotient[12] $\Lambda/_\sim$ is a compositional and $=_\beta$-invariant interpretation map. Note that this interpretation does not directly answer to the question of seeing a λ-term as a function, not even in the generalised sense of morphism. Actually, these two constructions are immediately seen inverse of each other (when we restrict the codomain of the interpretation map to its image). So to give a λ-theory exactly means to give a compositional, $=_\beta$-invariant and surjective interpretation map.

Typically, a λ-theory is given in a "syntactical" way (meaning by using properties of $\to_\beta$), while an interpretation map is given in a "semantical" way (meaning by interpreting λ-terms in an abstract structure). Remark that here again there is a clear analogy with what one does in polynomial algebra (but also group/ring/... theory).

In fact λ-theories play the same role of polynomial ideals (or normal subgroups, or ring ideals etc): they provide the notion of *compatible equivalence*, with respect to which the quotient maintains the same algebraic structure; in the case of the combinatory algebra $\Lambda/_{=_\beta}$, its quotients w.r.t. λ-theories are still combinatory algebras.

[12] The quotient of Λ by a λ-theory is called its *term-algebra*. Logicians and computer scientists also say "term-model", because historically it was thought of as syntactical *à la* Tarski model. I personally find it clearer in the way I presented it here.