# The algebraicity of the lambda-calculus

André Hirschowitz[1] and Marco Maggesi[2]

[1] Université de Nice – Sophia Antipolis
`http://math.unice.fr/~ah`
[2] Università degli Studi di Firenze
`http://www.math.unifi.it/~maggesi`

**Abstract.** We propose a new definition for abstract syntax (with binding constructions), and, accordingly, for initial semantics and algebraicity. Our definition is based on the notion of module over a monad and its companion notion of linearity. In our setting, we give a one-line definition of an untyped lambda-calculus. Among untyped lambda-calculi, the initial one, the pure untyped lambda-calculus, appears as defined by two algebraic constructions ($\mathsf{abs}$ and the unary application $\mathsf{app}_1$), together with two algebraic equations which are essentially the $\beta$ and $\eta$ rules.

## 1 Introduction

Although the lambda-calculus is by now seventy five years old, it still lacks a comprehensive definition. Its standard definition (see e.g. Wikipedia) is quite down-to-earth and requires tedious considerations on free versus bound variables and capture-avoiding substitution. Here we fill these gaps and discover the true nature of untyped lambda-calculi as follows:

- an untyped lambda-calculus is a functor $\mathsf{LC}$ from sets to sets: to a set $V$ (of free "variables") it assigns a set $\mathsf{LC}(V)$ of lambda-terms built out from $V$; this functor enjoys substitution, which turns it into a monad;
- this monad is equipped on one hand with the so-called abstraction: $\mathsf{abs}$ is a natural transformation $V \mapsto \mathsf{abs}_V : \mathsf{LC}(V^*) \longrightarrow \mathsf{LC}(V)$, where $V^*$ is obtained by adjoining an element to $V$; and on the other hand with the so-called unary application: $\mathsf{app}_1$ is a natural transformation $V \mapsto (\mathsf{app}_1)_V : \mathsf{LC}(V) \longrightarrow \mathsf{LC}(V^*)$;
- these natural transformations are compatible with substitution, more precisely they are (right-)$\mathsf{LC}$-module morphisms (see [HM06]): indeed, both $V \mapsto \mathsf{LC}(V)$ and $V \mapsto \mathsf{LC}(V^*)$ are (right) $\mathsf{LC}$-modules and we say that the latter is the derivative of the former;
- these two morphisms $\mathsf{abs}$ and $\mathsf{app}_1$ are inverse of each other, as expressed by the familiar $\beta$ and $\eta$ rules.

There is a natural category of untyped lambda-calculi, where our theorem asserts the existence of an initial object: the pure untyped lambda-calculus.

Summarizing, we have the one-line definitions:

– an untyped lambda-calculus is a monad over $\mathsf{Set}$ equipped with a module isomorphism to its derivative;
– the pure untyped lambda-calculus is the initial untyped lambda-calculus.

This looks like a new point of view (for the old point of view, see [Sel96]) concerning the algebraicity of the lambda-calculus, which appears as defined by two algebraic constructions, $\mathsf{abs}$ and $\mathsf{app}_1$, together with two algebraic equations, $\beta$ and $\eta$, expressing that the two constructions are inverse of each other.

The point of view proposed here can be accommodated at least to model the simply-typed lambda-calculus through a monad on the category $\mathsf{Set}/T$ where $T$ is the set of simple types (see [Zsi06]), or to model $\beta$-reduction through a monad on the category of partially ordered sets.

This paper is organized as follows. Section 2 gives a succinct account about modules over a monad. In section 3, we develop a simple first-order typed syntax adapted to our needs. A framework for abstract syntax with bindings is introduced in section 4. Section 5 develops the theory of equations for our notion of syntax and presents untyped lambda-calculus as an *algebraic theory* where the $\beta$ and $\eta$ rules appear as algebraic equations. The last section contains some pointers to and comparison with related works.

## 2 Modules over monads

We recall only the definition and some basic facts about (right) modules over a monad. See [HM06] for a more extensive introduction on this topic.

Let $\mathsf{C}$ be a category. A monad over $\mathsf{C}$ is a monoid in the category $\mathsf{C} \to \mathsf{C}$ of endofunctors of $\mathsf{C}$, i.e., a triple $R = \langle R, \mu, \eta \rangle$ given by a functor $R \colon \mathsf{C} \to \mathsf{C}$, and two natural transformations $\mu \colon R^2 \to R$ and $\eta \colon I \to R$ such that the following diagrams commute:

$$\begin{tikzcd} R^3 \arrow[r, "R\mu"] \arrow[d, "\mu R"'] & R^2 \arrow[d, "\mu"] \\ R^2 \arrow[r, "\mu"] & R \end{tikzcd} \qquad \begin{tikzcd} I \cdot R \arrow[r, "\eta R"] \arrow[dr, "1_R"'] & R^2 \arrow[d, "\mu"] & R \cdot I \arrow[l, "R\eta"'] \arrow[dl, "1_R"] \\ & R & \end{tikzcd}$$

Let $R$ be a monad over $\mathsf{C}$.

**Definition 1 (Right modules).** *A right $R$-module is given by a functor $M \colon \mathsf{C} \to \mathsf{D}$ equipped with a natural transformation $\rho \colon M \cdot R \to M$, called* action*, which is compatible with the monad composition:*

$$\begin{tikzcd} M \cdot R^2 \arrow[r, "M\mu"] \arrow[d, "\rho R"'] & M \cdot R \arrow[d, "\rho"] \\ M \cdot R \arrow[r, "\rho"] & M \end{tikzcd} \qquad \begin{tikzcd} M \cdot R \arrow[d, "\rho"'] & M \cdot I \arrow[l, "M\eta"'] \arrow[dl, "1_M"] \\ M & \end{tikzcd}$$

*We will refer to the category $\mathsf{D}$ as the* range *of $M$.*

*We say that a natural transformation of right $R$-modules $\tau\colon M \to N$ is* linear *if it is compatible with substitution:*

$$\begin{tikzcd} M \cdot R \arrow[r, "\tau R"] \arrow[d, "\rho_M"'] & N \cdot R \arrow[d, "\rho_N"] \\ M \arrow[r, "\tau"] & N \end{tikzcd}$$

*We take linear natural transformations as morphisms among right modules having the same range $D$. It can be easily verified that we obtain in this way a category that we denote* $\mathrm{Mod}^{\mathsf{D}}(R)$*.*

There is an obvious corresponding definition of left $R$-modules that we do not need to consider in this paper. From now on, we will write $R$-modules instead of right $R$-modules for brevity.

*Example 1.* Let us show some trivial examples of modules:

1. Every monad monad $R$ is a module over itself, which we call the *tautological* module.
2. For any functor $F\colon \mathsf{D} \to \mathsf{E}$ and any $R$-module $M\colon \mathsf{C} \to \mathsf{D}$, the composition $F \cdot M$ is a $R$-module (in the evident way).
3. For every object $W \in \mathsf{D}$ we denote by $\underline{\mathrm{W}}\colon \mathsf{C} \to \mathsf{D}$ the constant functor $\underline{\mathrm{W}} := X \mapsto W$. Then $\underline{\mathrm{W}}$ is trivially a $R$-module since $\underline{\mathrm{W}} = \underline{\mathrm{W}} \cdot R$.

Limits and colimits in the category of right modules can be constructed point-wise. In particular:

**Lemma 1 (Limits and colimits of modules).** *If* $\mathsf{D}$ *is complete (resp. cocomplete), then* $\mathrm{Mod}^{\mathsf{D}}(R)$ *is complete (resp. cocomplete).*

In particular, we will often make use of the fact that, if the range category $\mathsf{D}$ is cartesian, then the category $\mathrm{Mod}^{\mathsf{D}}(R)$ is also cartesian.

For our purposes, one important example of module is given by the following general construction. Let $\mathsf{D}$ be a category with finite colimits and a final object $*$ and consider the final functor $\underline{*}\colon \mathsf{C} \to \mathsf{D}$.

**Definition 2 (Derivation).** *For any $R$-module $M$ with range* $\mathsf{D}$*, the* derivative *of $M$ is the colimit $M' := M + \underline{*}$ of $M$ and $\underline{*}$. Derivation can be iterated, we denote by $M^{(k)}$ the $k$-th derivative of $M$.*

**Proposition 1.** *Derivation yields an endofunctor of* $\mathrm{Mod}^{\mathsf{D}}(R)$*. Moreover, if* $\mathsf{D}$ *is a cartesian category, derivation is a cartesian endofunctor of* $\mathrm{Mod}^{\mathsf{D}}(R)$*.*

In the case $\mathsf{C} = \mathsf{D} = \mathsf{Set}$, the functor $M'$ is given by $M' := X \mapsto M(X + *)$, where $X + *$ denotes the set obtained by adding a new point to $X$. Moreover, we have a natural *evaluation morphism*

$$\mathsf{eval}\colon M' \times R \longrightarrow M.$$

which is $R$-linear. This allows us to interpret the derivative $M'$ as the "module $M$ with one formal parameter added". Higher order derivatives have analogous morphisms (that we still denote with $\mathsf{eval}$)

$$\mathsf{eval}\colon M^{(b)} \times R^b \longrightarrow M.$$

where $\mathsf{eval}(t, m_1, \ldots, m_b) \in M(X)$ is obtained by substituting $m_1, \ldots, m_b \in R(X)$ in the successive stars of $t \in M^{(b)}(X) = M(X + * + \cdots + *)$.

We already introduced the category $\mathrm{Mod}^{\mathsf{D}}(R)$ of modules with fixed base $R$ and range $D$. It it often useful to consider a larger category which collects modules with different bases. To this end, we need first to introduce the notion of pull-back.

**Definition 3 (Pull-back).** *Let $f\colon R \to S$ be a morphism of monads and $M$ a $S$-module. The action*

$$R \cdot M \xrightarrow{fM} S \cdot M \xrightarrow{\rho} M$$

*defines a $R$-module which is called* pull-back *of $M$ along $f$ and noted $f^*M$. It can be easily verified that a $S$-linear natural transformation $g\colon M \to N$ is also a $R$-linear natural transformation $f^*g\colon f^*M \to f^*N$ and that $f^*\colon \mathrm{Mod}^{\mathsf{D}}(S) \to \mathrm{Mod}^{\mathsf{D}}(R)$ is a functor.*

It can be easily verified that pull-back is well-behaved with respect to many important constructions. In particular:

**Proposition 2.** *Pull-back commutes with products and with derivation.*

**Definition 4 (The large module category).** *We define the large module category* LMod *as follows:*

- *its objects are pairs $(R, M)$ of a monad $R$ and a $R$-module $M$.*
- *a morphism from $(R, M)$ to $(S, N)$ is a pair $(f, m)$ where $f\colon R \longrightarrow S$ is a morphism of monads, and $m\colon M \longrightarrow f^*N$ is a morphism of $R$-modules. The category* LMod *comes equipped with a forgetful functor to the category of monads and with derivation as an endofunctor.*

## 3 First-order graded syntax

In this section we expose a simple kind of first order syntax, "graded" over a fixed set $D$, especially tuned for our applications to higher-order syntax (where we will need only the case $D = \mathbb{N}$ though). We do not claim any originality of thought for the material in this section.

**Definition 5 ($D$-sets).** *We call $D$-set any family of sets indexed by $D$. We have a category of $D$-sets, where morphisms are applications preserving the degree.*

For each $D$-set $X$ and $d \in D$, we denote by $X_d$ the component of $X$ in degree $d$.

**Definition 6 ($D$-arities and $D$-signatures).** *A $D$-arity is a finite non-empty list of elements in $D$. We use a colon to single out this first element of the list. A signature is a family of arities, i.e., a pair $\Sigma = (O, \alpha)$ where $\alpha(o) = (b(o) : a(o))$ is an arity for every $o \in O$.*

**Definition 7 (Representation of a $D$-arity).** *Given a $D$-set $M$, we define a* representation *of the arity $\alpha = (b : a_1, \ldots, a_n)$ in $M$ to be an application*

$$r \colon \prod_{i=1}^{n} M_{a_i} \longrightarrow M_b.$$

*We also say that $r$ is a* construction *of arity $\alpha$ in $M$.*

**Definition 8 (Representation of a $D$-signature).** *A representation of the signature $\Sigma = (O, \alpha)$ in the $D$-set $M$, consists of, for each $o$ in $O$, a representation of the arity $\alpha(o)$ in $M$.*

**Definition 9 (The category of representations).** *Given a signature $\Sigma = (O, \alpha)$, we build the category $\mathrm{Grad}_D^{\Sigma}$ of representations of $\Sigma$ in $D$-sets as follows. Its objects are $D$-sets equipped with a representation of $\Sigma$. A morphism from $(M, r)$ to $(N, s)$ is a $D$-morphism from $M$ to $N$ compatible with the representations in the sense that, for each $o$ in $O$, with $\alpha(o) = (b : a_1, \ldots, a_n)$, the following diagram commutes:*

$$\begin{array}{ccc} \prod_{i=1}^{n} M_{a_i} & \longrightarrow & M_b \\ \big\downarrow & & \big\downarrow \\ \prod_{i=1}^{n} N_{a_i} & \longrightarrow & N_b \end{array} \tag{1}$$

*where the horizontal arrows come from the representations and the vertical arrows come from $f$.*

**Proposition 3.** *These morphisms, together with the obvious composition, turn $\mathrm{Grad}_D^{\Sigma}$ into a category.*

**Proposition 4.** *The category $\mathrm{Grad}_D^{\Sigma}$ has an initial object which we call the inductive $D$-set generated by $\Sigma$, and denote by $\hat{\Sigma}$.*

As usual, our "syntactic" object $\mathrm{Grad}_D^{\Sigma}$ comes with a corresponding recursion and induction principle which we avoid to state explicitly.

## 4 Higher-order untyped syntax

In the previous section we have considered constructions without bindings. Here we take $\mathbb{N}$-arities in order to manage constructions with bindings.

**Definition 10 (Arity).** *We define an* arity *to be a* $\mathbb{N}$*-arity, namely a non-empty list of integers. We say the arity is* raw *when the first element of the list is zero. Given an arity* $(b : a_1, \ldots, a_n)$*, we denote by* $\mathrm{raw}(a)$ *the (raw) arity* $(0 : a_1, \ldots, a_n, 0^b)$*, where* $0^b$ *stands for a list of* $b$ *zeros.*

The difference with the definition in [FPT99] is that our arity provides one more integer whose intended meaning is the number of extra formal arguments of the output of the operation.

*Example 2.* From this point of view, we can imagine three different arities for the application of the lambda-calculus: the classical arity is $(0 : 0, 0)$, and the other two are $(1 : 0)$ and $(2 :)$.

**Definition 11 (Signatures).** *We define a (binding) signature* $\Sigma = (O, \alpha)$ *to be a family of arities* $\alpha \colon O \to \mathbb{N} \times \mathbb{N}^*$*. Here* $\mathbb{N}^*$ *stands for the set of lists of integers. A signature is said to be raw if it consists of raw arities. For a signature* $\Sigma = (O, \alpha)$*, we denote by* $\mathrm{raw}(\Sigma)$ *the raw signature* $(O, o \mapsto \mathrm{raw}(\alpha(o)))$*.*

**Definition 12 (Representation of an arity).** *Given a monad* $M$ *over* $\mathsf{Set}$*, we define a representation of the arity* $\alpha = (b : a_1, \ldots, a_n)$ *in* $M$ *to be a module morphism*

$$r \colon \prod_{i=1}^{n} M^{(a_i)} \longrightarrow M^{(b)}.$$

*We also say that* $r$ *is a* construction *of arity* $\alpha$ *in* $M$*. In case* $b = 0$ *we say that* $r$ *is a raw construction in* $M$*.*

**Definition 13 (Flattening a representation of an arity).** *Given a monad* $M$ *and a representation* $r$ *of the arity* $\alpha = (b : a_1, \ldots, a_n)$ *in* $M$*, we define the natural transformation*

$$\mathrm{raw}(r) \colon \prod M^{(a_i)} \times M^b \longrightarrow M$$

*as the composition*

$$\prod M^{(a_i)} \times M^b \xrightarrow{r \times M^b} M^{(b)} \times M^b \xrightarrow{\mathsf{eval}} M$$

*It is easily checked that* $\mathrm{raw}(r)$ *is a representation of* $\mathrm{raw}(\alpha)$*.*

**Proposition 5 (Flattening is bijective for arities).** *Given a monad* $M$ *and an arity* $\alpha = (b : a_1, \ldots, a_n)$ *the raw map defined above defines a bijection from the set of representations of* $\alpha$ *in* $M$ *to the set of representations of* $\mathrm{raw}(\alpha)$ *in* $M$*.*

*Example 3.* A representation of $(2 :)$ in $\mathsf{LC}$ is given by the $\mathsf{app}_0 \colon \mathsf{LC}^{(2)}$ construction. The associated representation of $\mathrm{raw}(2 :) = (0 : 0, 0)$ is the usual $\mathsf{app}$.

**Definition 14 (Representation of a signature).** *A representation of the signature* $\Sigma = (O, \alpha)$ *in the monad* $M$*, consists of, for each* $o$ *in* $O$*, a representation of* $\alpha(o)$ *in* $M$*.*

**Definition 15 (Flattening a representation of a signature).** *Given a monad $M$ and a representation $r$ of the signature $\Sigma = (O, \alpha)$ we define* $\mathrm{raw}(r)$ *to be the family $o \mapsto \mathrm{raw}(r(o))$ which is a representation of* $\mathrm{raw}(\Sigma)$.

**Proposition 6 (Flattening is bijective for signatures).** *Given a monad $M$ and a signature $\Sigma = (O, \alpha)$ the raw map defined above defines a bijection from the set of representations of $\Sigma$ in $M$ to the set of representations of* $\mathrm{raw}(\Sigma)$ *in $M$.*

**Definition 16 (The category of representations).** *Given a signature $\Sigma = (O, \alpha)$, we build the category* $\mathrm{Mon}^{\Sigma}$ *of representations of $\Sigma$ as follows. Its objects are monads equipped with a representation of $\Sigma$. A morphism from $(M, r)$ to $(N, s)$ is a morphism from $M$ to $N$ compatible with the representations in the sense that, for each $o$ in $O$, $\alpha(o) = (b : a_1, \ldots, a_n)$, the following diagram commutes:*

$$\begin{array}{ccc} \prod_{i=1}^{n} M^{(a_i)} & \longrightarrow & M^{(b)} \\ \big\downarrow & & \big\downarrow \\ \prod_{i=1}^{n} f^* N^{(a_i)} & \longrightarrow & f^* N^{(b)} \end{array} \tag{2}$$

*where the horizontal arrows come from the representations and the vertical arrows come from $f$ (it is used here that $f^*$ commutes with derivation and products).*

**Proposition 7.** *These morphisms, together with the obvious composition, turn* $\mathrm{Mon}^{\Sigma}$ *into a category which comes equipped with a forgetful functor to the category of monads.*

**Proposition 8.** *Given a signature $\Sigma$, the raw construction extends as a functor*

$$\mathrm{Mon}^{\Sigma} \longrightarrow \mathrm{Mon}^{\mathrm{raw}\,\Sigma}$$

*from the category or representations of $\Sigma$ to the category of representations of* $\mathrm{raw}(\Sigma)$*. Furthermore, this functor is an isomorphism of categories.*

**Definition 17 (Deriving arities and signatures).** *Given an arity $\alpha := (b : a)$, we define its derivative $\alpha'$ by adding one to each element of the list. By iteration, we define similarly the $n$-th derivative ($n \geq 0$) of an arity and accordingly of a signature.*

*Remark 1.* Since derivation is a functor, to every construction $c$ of arity $\alpha$ in a monad $M$, there is an induced construction $c'$ of arity $\alpha'$ and, more generally, a construction $c^{(n)}$ for each $n \geq 0$. Similarly, a representation of the signature $\Sigma$, induces a family of representations in $\mathbb{N}$-sets of the signatures $\Sigma^{(n)}$.

Let $M$ be a monad, $r$ a $\Sigma$-representation in $M$ with $\Sigma := (O, \alpha)$. The $\Sigma$-representation $r$ induces a family of representation of $\mathbb{N}$-sets as follows. First consider, for each set $X$, the associated $\mathbb{N}$-set $M_X := i \mapsto M^{(i)}(X)$. Then, for each $X$, we have several constructions induced on $M_X$ by $r$ and the structure of $M^{(i)}$. First of all, remark 1 gives us a construction of arity $\alpha(o)^{(n)}$ for each $o \in O$ and $n \geq 0$. Next, the $i$ inclusions $* \to M^{(i)}$ provide a construction for all $i \geq 0$ and $j \in [1, \ldots, i]$ of arity $(i :)$. Finally, the unity $\eta$ of $M$, with all its derivatives, gives a construction of arity $(n :)$ for each $x \in X$ and $n \in \mathbb{N}$. Putting all these constructions together, we are led to consider, for every set $X$, the (raw) $\mathbb{N}$-signature $\tilde{\Sigma}_X$ associated to $\Sigma$ consisting of the following three families of arities:

1. for each non-negative integer $n$, $\Sigma^{(n)}$;
2. for each integer $n$, a family indexed by $X$ of arities equal to $(n :)$;
3. for each integer $n$, a family indexed $[1, \ldots, n]$ of arities $(n :)$.

**Theorem 1.** *The category* $\mathrm{Mon}^\Sigma$ *has an initial object, which we call the syntactic monad generated by* $\Sigma$*, and denote by* $\hat{\Sigma}$*. Furthermore, for each set* $X$*, the* $\mathbb{N}$*-set* $n \mapsto \hat{\Sigma}^{(n)}(X)$ *is the inductive* $\mathbb{N}$*-set generated by the* $\mathbb{N}$*-signature* $\tilde{\Sigma}_X$*.*

*Proof.* By the previous proposition, we may suppose that $\Sigma$ is raw. For each set $X$, let $T_X$ be the inductive $\mathbb{N}$-set generated by $\tilde{\Sigma}_X$. The statement gives us for $\hat{\Sigma}$ the formula

$$\hat{\Sigma} \colon X \mapsto T_X$$

which has a natural structure of monad and admits an obvious representation of $\Sigma$. The remaining verifications are not difficult.

## 5 The category of half-equations

In this section, we are given a signature $\Sigma$, with associated syntax $\hat{\Sigma}$, and we build the category where will live our (algebraic) equations.

**Definition 18 (The category of half-equations).** *We define a* $\Sigma$*-module* $U$ *to be a functor from the category of representations of* $\Sigma$ *to the category* LMod *commuting to the forgetful functors to the category* Mon *of monads.*

$$\begin{tikzcd} \mathrm{Mon}^\Sigma \arrow[rr, "U"] \arrow[dr] & & \mathrm{LMod} \arrow[dl] \\ & \mathrm{Mon} & \end{tikzcd}$$

*We define a morphism of* $\Sigma$*-modules to be a natural transformation which becomes the identity when composed with the forgetful functor. We call these morphisms "half-equations".*

*These definitions yield a category which we call the category of half-equations (for* $\Sigma$*).*

*Example 4.* We denote by $T$ the tautological $\Sigma$-module, which assigns to a representation in $M$ the tautological module over $M$. Since derivation is an endofunctor in LMod, it acts on $\Sigma$-modules. So we also have a family of $\Sigma$-modules $T^{(n)}$.

*Example 5.* To each construction $c \in \Sigma$ with arity $(b : a_1, \ldots, a_n)$ is associated a morphism $\mathsf{rep}(c) \colon \prod_{i=1}^{n} T^{(a_i)} \longrightarrow T^{(b)}$ in the obvious way.

**Proposition 9.** *The category of half-equations is cartesian.*

*Example 6.* To each application $f \colon [1, \ldots, p] \longrightarrow [1, \ldots, q]$ is associated a half-equation: $f_* \colon T^{(p)} \longrightarrow T^{(q)}$ (which could be called renaming along $f$).

**Definition 19 (Equations).** *We define an equation for $\Sigma$ to be a pair of half-equations with common source and target. We also write $e_1 = e_2$ for the equation $(e_1, e_2)$.*

*Example 7.* In case $\Sigma$ consists of the two constructions $\mathsf{abs}$ and $\mathsf{app}_1$ with respective arities $(0 : 1)$ and $(1 : 0)$, the $\beta$ equation is $\mathsf{app}_1 \circ \mathsf{abs} = \mathrm{Id}_{T^{(1)}}$, while the $\eta$ equation is $\mathsf{abs} \circ \mathsf{app}_1 = \mathrm{Id}_T$.

**Definition 20 (Satisfying equations).** *We say that a representation $r$ of $\Sigma$ in a monad $M$ satisfies the equation $e_1 = e_2$ if $e_1(r) = e_2(r)$. If $E$ is a set of equations for $\Sigma$, we say that a representation $r$ of $\Sigma$ in a monad $M$ satisfies $E$ (or is a representation of $(\Sigma, E)$) if it satisfies each equation in $E$. We define the category of representations of $(\Sigma, E)$ to be the full subcategory in the category of representations of $\Sigma$ whose objects are representations of $(\Sigma, E)$.*

**Theorem 2 (Initial representation of $(\Sigma, E)$).** *Given a set $E$ of equations for $\Sigma$, the category of representations of $(\Sigma, E)$ has an initial object.*

*Proof.* We build the monad $S$ for our initial representation as a quotient of $\hat{\Sigma}$ (see theorem 1). For each set $X$, we define an equivalence relation $r_X$ on $\hat{\Sigma}(X)$ as follows: $r_X(a, b)$ means that for any representation $\rho$ of $(\Sigma, E)$ in a monad $M$, $i_X(a)$ equals $i_X(b)$, where $i \colon \hat{\Sigma} \longrightarrow M$ is the initial functor associated to $\rho$. We check easily that this is an equivalence relation, that the corresponding collection of quotients inherits the structure of monad, and that this quotient monad has the required universal property.

**Definition 21 (Algebraic half-equations).** *We define the category of algebraic half-equations to be the cartesian subcategory in the category of half-equations generated by*

1. *the tautological $\Sigma$-module,*
2. *derivation,*
3. *renamings,*
4. *half-equations associated to constructions in $\Sigma$.*

**Definition 22 (Algebraic equations).** *We say that an equation for $\Sigma$ is algebraic if its two halves are.*

**Definition 23 (Algebraic theories).** *We call algebraic theory any initial representation of a pair $(\Sigma, E)$, where $E$ is a set of algebraic equations for $\Sigma$.*

*Example 8.* The untyped lambda-calculus is an algebraic theory.

## 6 Related works

The idea that the notion of monad is suited for modelling substitution concerning syntax (and semantics) has been retained by many recent contributions concerned with syntax (see e.g. [BP99] [GU03] [MU04]) although some other settings have been considered. Notably in [FPT99] the authors work within a setting roughly based on operads (although they do not write this word down; the definition of operad is on Wikipedia; operads and monads are not too far from each other). Our main specificity here is the systematic use of the observation that the natural transformations we deal with are linear with respect to natural structures of module (a form of linearity had already been observed, in the operadic setting, see [FT01], section 4) .

The signatures we consider here are slightly more general than the signatures in [FPT99], which allows us to give a signature to our $app_1$. Even the very general and abstract notion of signature coined in [MU04] can encompass can encompass our notion only in a tricky way. On the other hand, our signatures "reduce", in the sense we have explained, to those in [FPT99].

An other way to establish the algebricity of lambda-calculus has been proposed in [Sel96] through a detour via combinators. One advantage of our approach is that it prospects a general picture for the treatment of algebraic equations.

## References


[BP99] Richard Bird and Ross Paterson, *Generalised folds for nested datatypes*, Formal Aspects of Computing **11** (1999), no. 2, 200–222.

[FPT99] Marcelo Fiore, Gordon Plotkin, and Daniele Turi, *Abstract syntax and variable binding*, LICS '99: Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (Washington, DC, USA), IEEE Computer Society, 1999, p. 193.

[FT01] Marcelo P. Fiore and Daniele Turi, *Semantics of name and value passing*, Logic in Computer Science, 2001, pp. 93–104.

[GU03] Neil Ghani and Tarmo Uustalu, *Explicit substitutions and higher-order syntax*, MERLIN '03: Proceedings of the 2003 ACM SIGPLAN workshop on Mechanized reasoning about languages with variable binding (New York, NY, USA), ACM Press, 2003, pp. 1–7.

[HM06] Andre Hirschowitz and Marco Maggesi, *Modules over monads and linearity*, 2006, `http://www.math.unifi.it/~maggesi/`.

[MU04] Ralph Matthes and Tarmo Uustalu, *Substitution in non-wellfounded syntax with variable binding*, Theor. Comput. Sci. **327** (2004), no. 1-2, 155–174.

[Sel96] P. Selinger, *The lambda calculus is algebraic*, 1996.

[Zsi06] Julianna Zsidó, *Le lambda calcul vu comme monade initiale*, Master's thesis, Université de Nice – Laboratoire J. A. Dieudonné, 2005/06, Mémoire de Recherche – master 2.