Accepted for publication in Math. Struct. in Comp. Science. Revised version.

A Typed, Algebraic, Computational Lambda-Calculus

Benoît Valiron

LIPN – UMR 7030 CNRS – Université Paris 13
99 av. J-B Clément,
F-93430 Villetaneuse, France

`benoit.valiron@monoidal.net`

Received October 2011

Lambda-calculi with vectorial structures have been studied in various ways, but their semantics remain mostly untouched. The main contribution of this paper is to provide a categorical framework for the semantics of such algebraic lambda-calculi. We first develop a categorical analysis of a general simply-typed lambda-calculus endowed with a structure of module. We study the problems arising from the addition of a fixed point combinator and show how to modify the equational theory to solve them. The categorical analysis carries nicely over to the modified language. We provide various concrete models, both for the case without fixpoints and the case with fixpoints.
Keywords: algebraic lambda-calculus, module over ring and semi-ring, fixpoints, semantics, equational theory, categorical models.

Contents

1. Introduction

The term “algebraic lambda-calculus” comes from a line of work (Breazu-Tannen and Gallier, 1991; Blanqui et al., 1999; Barbanera and Fernández, 1993) which focuses on general algebraic rewrite systems and studies the conditions needed for obtaining properties such as confluence or strong normalization. In this paper, we are concerned with the particular algebraic structure of module over a lambda-calculus, and we shall use the term “algebraic” or “vectorial” to refer to this particular structure. Vectorial lambda-calculi have at least two distinct origins. The first one is the calculus of (Vaux, 2009), building up upon the work of (Ehrhard and Regnier, 2003). The goal here is to capture a notion of differentiation within lambda-calculus. The notion of algebraic lambda-calculus also arises in the work of (Arrighi and Dowek, 2008) where a lambda-calculus oriented towards quantum computation is defined in the style of (van Tonder, 2004).

Both (Arrighi and Dowek, 2008) and (Vaux, 2009) are concerned with a lambda-calculus endowed with a structure of vector space. They both acknowledge the fact that for an untyped lambda-calculus, a naive rewrite system renders the language inconsistent, as any term can be made equal to the zero of the vectorial space of terms. However, coming from different backgrounds, they provide different solutions to the problem. In (Arrighi and Dowek, 2008), the rewrite system is restrained in order to avoid unwanted equalities of terms. In (Vaux, 2009), the rewrite system is untouched, but the scalars over which the vectorial structure is built are made into a positive semiring with particular properties, making the system consistent. Finally, (Arrighi and Díaz-Caro, 2009) shows that a type system enforcing strong normalization can also be a mean of solving the problem.

In this paper, we turn to the yet untouched question of the semantics of lambda-calculi endowed with a structure of vector space (or more generally, a structure of module). We are not concerned with the question of a rewrite system, but only with the equational theory that is required to make the system consistent. In the following, we recall what are the problems occurring while defining a naive equational theory for a vectorial lambda-calculus.

Table 1. *Naive equational system.*

	Algebraic rules	
$0 \cdot s \simeq_{ax} \mathbf{0}$	$\mathbf{0} + s \simeq_{ax} s$	$\alpha \cdot (\beta \cdot s) \simeq_{ax} (\alpha\beta) \cdot s$
$\alpha \cdot \mathbf{0} \simeq_{ax} \mathbf{0}$	$1 \cdot s \simeq_{ax} s$	$\alpha \cdot (s + t) \simeq_{ax} \alpha \cdot s + \alpha \cdot t$
$\alpha \cdot s + \beta \cdot s \simeq_{ax} (\alpha + \beta) \cdot s$	$r + s \simeq_{ax} s + r$	$r + (s + t) \simeq_{ax} (r + s) + t$
	Rules for call-by-value	
$(s + t)r \simeq_{ax} sr + tr$	$(\alpha \cdot s)r \simeq_{ax} \alpha \cdot (sr)$	$\mathbf{0}r \simeq_{ax} \mathbf{0}$
$r(s + t) \simeq_{ax} rs + rt$	$r(\alpha \cdot s) \simeq_{ax} \alpha \cdot (rs)$	$r\mathbf{0} \simeq_{ax} \mathbf{0}$
	$(\lambda x.s)v \simeq_{ax} s[x \leftarrow v]$	
	Rules for call-by-name	
$(s + t)r \simeq_{ax} sr + tr$	$(\alpha \cdot s)r \simeq_{ax} \alpha \cdot (sr)$	$\mathbf{0}r \simeq_{ax} \mathbf{0}$
$\lambda x.(s + t) \simeq_{ax} \lambda x.s + \lambda x.t$	$\lambda x.(\alpha \cdot s) \simeq_{ax} \alpha \cdot \lambda x.s$	$\lambda x.\mathbf{0} \simeq_{ax} \mathbf{0}$
	$(\lambda x.s)t \simeq_{ax} s[x \leftarrow t]$	

1.1. *An untyped calculus*

Consider a ring $(\mathcal{A}, +, 0, \times, 1)$. Elements of $\mathcal{A}$ are called *scalars.* A lambda-calculus together with a vectorial structure over $\mathcal{A}$ should at least contain the terms

$$s, t \quad ::= \quad x \mid \lambda x.s \mid st \mid s + t \mid \alpha \cdot s \mid \mathbf{0},$$

where α ranges over $\mathcal{A}$, and where x ranges over a fixed set of term variables.

1.2. *Naive equational system*

In order to give some meaning to the language, we equip it with an equivalence $\simeq_{ax}$ on terms, called *axiomatic equivalence.*

The equivalence should make the set of terms into a module over a ring $\mathcal{A}$ with the term $\mathbf{0}$ as unit of the addition. In particular, the addition on terms should be commutative and associative, and the terms $t - t$ and $0 \cdot t$ should equate the term $\mathbf{0}$. This is described in the algebraic set of rules in Table 1.

Now, we need to set some distributivity laws on the term constructs and to say how the lambda-abstraction and application interact with each other. For reasons similar to the ones appearing in probabilistic languages, we cannot both have distributivity on the left side of the application and a general substitution:

$$(\lambda x.(fx)x)(s + t) \simeq_{ax} (f(s + t))(s + t) \simeq_{ax} (fs)s + (fs)t + (ft)s + (ft)t$$

cannot be equated with

$$(\lambda x.(fx)x)(s + t) \simeq_{ax} (\lambda x.(fx)x)s + (\lambda x.fxx)t \simeq_{ax} (fs)s + (ft)t$$

in general. If f stands for the boolean operation XOR and s and t respectively for the value *true* and the value *false*, the two paths cannot be merged: the former corresponds to the sum of true and false and the latter is equal to true.

We therefore have to decide on a strategy for substitution. The usual call-by-value and call-by-name strategies (Arrighi and Dowek, 2008; Vaux, 2009) have been studied in the literature for vectorial lambda-calculi.

Call-by-value. In this case, the application is distributive on both side, and the lambda-abstraction is not distributive at all. There is a notion of value: a value is either a term variable x or a lambda-abstraction $\lambda x.s$, for s any term. The equivalence on terms for call-by-value is defined in Table 1, together with some congruence rules. In Table 1, the term v stands for a value, and the notation $s[x \leftarrow t]$ stands for the term substitution of all the free instances of x by t in s.

Call-by-name. In call-by-name, an argument is substituted in the body of a function without being evaluated. In our case, this is to say that the application is distributive on the left but not on the right. For consistency, the lambda-abstraction is also distributive. The rules are found in Table 1.

1.3. *Breaking consistency*

Although the set of requirements for call-by-name and call-by-value looks reasonable, as was shown in (Arrighi and Dowek, 2008), the equational system is not sound. Indeed, given any term b where x is not free, one can construct a diverging term $Y_b = (\lambda x.(xx + b))(\lambda x.(xx + b))$ verifying the equation

$$Y_b \simeq_{ax} Y_b + b \tag{1}$$

both in call-by-name and in call-by-value, rendering the system inconsistent as enlightened in the following sequence of equalities:

$$\mathbf{0} \simeq_{ax} Y_b - Y_b \simeq_{ax} (Y_b + b) - Y_b \simeq_{ax} b + (Y_b - Y_b) \simeq_{ax} b. \tag{2}$$

This shows that any term can be equated to $\mathbf{0}$.

Solutions in the literature. There are various solutions to this problem. The most obvious one is to forbid diverging terms, either by modifying the rewrite system so that some terms do not rewrite anymore (Arrighi and Dowek, 2008) or by adding a type system to the language, as was done in (Vaux, 2009; Arrighi and Díaz-Caro, 2009) so that the diverging terms are not allowed at all. A third option (Vaux, 2009) is to work with positive semirings instead of rings. In this case, the addition does not have an inverse anymore, and it solves the inconsistency.

This is not the choice of this paper, where we want to be able to work with a ring and nonetheless be able to have fixpoints.

1.4. *Plan of the paper and highlight of contributions*

In this paper, we are interested in the axiomatic semantics of the vectorial lambda-calculus, and in the interpretation of divergence in this context. This novel analysis is general enough to be able to capture distinct notions of convergence, as enlightened in Section 3.3.

In Section 2, we present a simply-typed lambda-calculus generalizing the call-by-value and call-by-name lambda-calculi sketched in 1.2. In Section 2.1 we give the equational theory associated to the language. We provide a categorical model in Section 2.2 and prove that the interpretation of the axiomatic description is sound (Theorem 2.28) and complete (Theorem 2.37). In Section 2.3, we give various concrete models, effectively showing that the equational description is consistent.

In Section 3, we turn to the question of how to add fixpoints to the language. We said in Section 1.3 that adding a fixpoint combinator breaks the equational theory by making all terms equal to the zero of the module of terms. In Sections 3.1 and 3.2 we perform an analysis of the problem in the light of the developed semantics. We come to the conclusion that the only problematic rule is the one stating that $0 \cdot s = \mathbf{0}$ and we adjust the semantics accordingly. We devote Section 3.3 to the construction of a concrete instance of the modified categorical structure. In Sections 3.4 and 3.5, we adjust the algebraic computational lambda-calculus and its equational theory to support fixpoints. We show in Section 3.6 how various terms can be interpreted in the concrete model of Section 3.3 and we state in Section 3.7 the consistency of the resulting system.

Finally, in Section 4, we discuss various issues and the relation to other works in the literature. We conclude the paper in Section 5.

A preliminary version of this work appeared in (Valiron, 2010).

2. A simply-typed lambda-calculus

The problem occurring in Section 1.3 is due to the possibility of constructing diverging terms. In this section we define a simply-typed, vectorial lambda-calculus with two distinct lambda-abstractions for being able to encode the two behaviors described in Section 1.2: one lambda-abstraction will be distributive over the vectorial structure while the other will not be. We equip this language with an axiomatic equivalence relation, then develop the categorical analysis of the generated equational theory.

Definition 2.1. We suppose the existence of a ring $\mathcal{A}$, called the *ring of scalars*. In particular, the scalars can be summed and multiplied. Any scalar α admits an inverse $-\alpha$ with respect to the addition. The sum admits a unit 0 and the multiplication a unit 1. A simply-typed, call-by-value, vectorial lambda-calculus called the *computational algebraic lambda-calculus* is constructed as follows. Types are of the form

$$A, B \quad ::= \quad \iota \mid A \to B \mid A \Rightarrow B \mid A \times B \mid \top,$$

where ι ranges over a set of type constants. Terms are implicitly typed and come in two flavors:

$$\begin{aligned} s, t \quad &::= \quad x \mid \lambda x.s \mid \Lambda x.s \mid st \mid \langle\, s, t \,\rangle \mid \pi_1(s) \mid \pi_2(s) \mid \ * \ \mid s + t \mid \alpha \cdot s \mid \mathbf{0}, \\ u, v \quad &::= \quad x \mid \Lambda x.s \mid \langle\, u, v \,\rangle \mid \pi_1(u) \mid \pi_2(u) \mid *, \end{aligned}$$

Table 2. *Typing rules.*

$$\frac{}{\Delta, x : A \vdash x : A}\ (id) \qquad \frac{\Delta, x : A \vdash s : B}{\Delta \vdash \Lambda x.s : A \Rightarrow B}\ (\Lambda) \qquad \frac{\Delta, x : A \vdash s : B}{\Delta \vdash \lambda x.s : A \to B}\ (\lambda)$$

$$\frac{\Delta \vdash s : A \Rightarrow B \quad \Delta \vdash t : A}{\Delta \vdash st : B}\ (app_\Lambda) \qquad \frac{\Delta \vdash s : A \to B \quad \Delta \vdash t : A}{\Delta \vdash st : B}\ (app_\lambda)$$

$$\frac{}{\Delta \vdash * : \top}\ (*) \qquad \frac{\Delta \vdash s : A_1 \times A_2}{\Delta \vdash \pi_i s : A_i}\ (\pi_i) \qquad \frac{\Delta \vdash s : A \quad \Delta \vdash t : B}{\Delta \vdash \langle\, s, t \,\rangle : A \times B}\ (\times)$$

$$\frac{}{\Delta \vdash \mathbf{0} : A}\ (\mathbf{0}) \qquad \frac{\Delta \vdash s : A}{\Delta \vdash \alpha \cdot s : A}\ (\alpha) \qquad \frac{\Delta \vdash s : A \quad \Delta \vdash s : A}{\Delta \vdash s + t : A}\ (+)$$

where $\alpha \in \mathcal{A}$. Terms of the form s, t are called *computations* and terms of the form u, v are called *base terms*. The terms $\Lambda x.s$ and $\lambda x.s$ are two lambda-abstractions whose difference lies in their behavior with respect to the module structure: the latter is distributive while the former is not. Using a standard notation, we shall use the notation $\lambda x_1 \dots x_n.s$ in place of $\lambda x_1.\lambda x_2. \dots \lambda x_n.s$.

We define the notions of typing context Δ and of typing derivation $\Delta \vdash s : A$ in the usual way (see e.g. Pierce, 2002). Terms are considered up to α-equivalence, and valid typing derivations are built using the rules of Table 2.

Lemma 2.2. Any valid typing derivation $\Delta \vdash s : A$ has a unique typing tree.

Proof. The proof is done by induction on the structure of s using the fact that the term s is implicitly typed: each subterm has a fixed type, and for each possibility only one typing rule can be applied. □

Lemma 2.3. Let $\Delta \vdash s : A$ be a valid typing judgment and let $x : B$ be a variable not belonging to Δ. Then $\Delta, x : B \vdash s : A$ is also valid.

Proof. Proof by induction on the typing derivation of $\Delta \vdash s : A$. □

Lemma 2.4. If u and v are any base terms, then $u[x \leftarrow v]$ is also a base term.

Proof. Proof by induction on the structure of u. □

Lemma 2.5 (Substitution). Let $\Delta \vdash v : A$ and $\Delta, x : A \vdash s : B$ be two valid typing derivations, where v is a base term. Then $\Delta \vdash s[x \leftarrow v] : B$ is a valid typing derivation.

Proof. By structural induction on the typing derivation of $\Delta, x : A \vdash s : B$. □

2.1. *Equational theory*

We equip the language with an axiomatic equivalence relation similar to the one in Table 1. The relation is augmented with the rules taking into account the new term constructs for the product.

Definition 2.6. Given a relation R on terms, we say that it is a *congruent relation* if for all pairs $(s, s'), (t, t') \in R$, the pairs (st, st'), $(st, s't)$, $(s + t, s + t')$, $(s + t, s' + t)$,

Table 3. *Axiomatic equivalence relation: Algebraic rules.*

(3)	$0 \cdot s \simeq_{ax} \mathbf{0}$	$s + \mathbf{0} \simeq_{ax} s$
	$1 \cdot s \simeq_{ax} s$	$\alpha \cdot s + \alpha \cdot t \simeq_{ax} \alpha \cdot (s + t)$
	$\alpha \cdot s + \beta \cdot s \simeq_{ax} (\alpha + \beta) \cdot s$	$(r + s) + t \simeq_{ax} r + (s + t)$
	$\alpha \cdot (\beta \cdot s) \simeq_{ax} (\alpha\beta) \cdot s$	$s + t \simeq_{ax} t + s$

Table 4. *Axiomatic equivalence relation: Distributivity rules.*

(4)	$\langle\, r + s, t \,\rangle \simeq_{ax} \langle\, r, t \,\rangle + \langle\, s, t \,\rangle$	$\pi_1(s + t) \simeq_{ax} \pi_1(s) + \pi_1(t)$	(5)
(6)	$\langle\, r, s + t \,\rangle \simeq_{ax} \langle\, r, s \,\rangle + \langle\, r, t \,\rangle$	$\pi_2(s + t) \simeq_{ax} \pi_2(s) + \pi_2(t)$	(7)
(8)	$\langle\, \alpha \cdot s, t \,\rangle \simeq_{ax} \alpha \cdot \langle\, s, t \,\rangle$	$\pi_1(\alpha \cdot s) \simeq_{ax} \alpha \cdot \pi_1(s)$	(9)
(10)	$\langle\, s, \alpha \cdot t \,\rangle \simeq_{ax} \alpha \cdot \langle\, s, t \,\rangle$	$\pi_2(\alpha \cdot s) \simeq_{ax} \alpha \cdot \pi_2(s)$	(11)
(12)	$\langle\, \mathbf{0}, t \,\rangle \simeq_{ax} \mathbf{0}$	$\pi_1(\mathbf{0}) \simeq_{ax} \mathbf{0}$	(13)
(14)	$\langle\, t, \mathbf{0} \,\rangle \simeq_{ax} \mathbf{0}$	$\pi_2(\mathbf{0}) \simeq_{ax} \mathbf{0}$	(15)
(16)	$(r + s)t \simeq_{ax} rt + st$	$(\alpha \cdot s)t \simeq_{ax} \alpha \cdot (st)$	(17)
(18)	$r(s + t) \simeq_{ax} rs + rt$	$\mathbf{0}t \simeq_{ax} \mathbf{0}$	(19)
(20)	$s(\alpha \cdot t) \simeq_{ax} \alpha \cdot (st)$	$t\mathbf{0} \simeq_{ax} \mathbf{0}$	(21)

(22) $\lambda x.(s + t) \simeq_{ax} \lambda x.s + \lambda x.t$ (23) $\lambda x.(\alpha \cdot s) \simeq_{ax} \alpha \cdot \lambda x.s$ (24) $\lambda x.\mathbf{0} \simeq_{ax} \mathbf{0}$

Table 5. *Axiomatic equivalence relation: computational rules.*

(25)	$\pi_1\langle\, u, v \,\rangle \simeq_{ax} u$	$(\lambda x.s)u \simeq_{ax} s[x \leftarrow u]$	(26)
(27)	$\pi_2\langle\, u, v \,\rangle \simeq_{ax} v$	$(\Lambda x.s)t \simeq_{ax} (\lambda x.s)t$	(28)
(29)	$\langle\, \pi_1(u), \pi_2(u) \,\rangle \simeq_{ax} u$	$\lambda x.(sx) \simeq_{ax} s$	(30)
(31)	$* \simeq_{ax} u^{\top}$	$\Lambda x.(ux) \simeq_{ax} u$	(32)
(33)	$(\lambda x.x)s \simeq_{ax} s$	$(\lambda x.(\lambda y.s))t \simeq_{ax} \lambda y.((\lambda x.s)t)$	(34)
(35)	$(\lambda x^{A \to B}.xt)s \simeq_{ax} st$	$(\lambda y.(\lambda x.(\lambda z.r)\langle\, x, y \,\rangle)s)t \simeq_{ax} (\lambda z.r)\langle\, s, t \,\rangle$	(36)
(37)	$(\lambda x.r)((\lambda y.s)t) \simeq_{ax} (\lambda y.(\lambda x.r)s)t$	$(\lambda x.(\lambda y.r)s)t \simeq_{ax} (\lambda y.(\lambda x.r)t)s$	(38)
(39)	$(\lambda x.\pi_1(x))s \simeq_{ax} \pi_1(s)$	$(\lambda x.\pi_2(x))s \simeq_{ax} \pi_2(s)$	(40)

$(\langle\, s,t \,\rangle, \langle\, s,t' \,\rangle)$, $(\langle\, s,t \,\rangle, \langle\, s',t \,\rangle)$, $(\pi_2 s, \pi_2 s')$, $(\pi_1 s, \pi_1 s')$, $(\alpha \cdot s, \alpha \cdot s')$, $(\lambda x.s, \lambda x.s')$ and $(\Lambda x.s, \Lambda x.s')$ are in R.

Definition 2.7. We define an equivalence relation $\simeq_{ax}$ on terms as the smallest congruent equivalence relation, closed under α-equivalence and the equations of Tables 3, 4 and 5. Two valid typing judgments $\Delta \vdash s, t : A$ are *axiomatically equivalent*, written $\Delta \vdash s \simeq_{ax} t : A$, if $s \simeq_{ax} t$ is provable.

Lemma 2.8. If u is a base term and if $u \simeq_{ax} v$ then v is also a base term.

Proof. By structural induction on the derivation of $u \simeq_{ax} v$. □

2.2. *Categorical model*

We now turn to the question of the structure of this equational theory. It is composed of various pieces: a notion of module, a distinction between base terms and computations, and two notions of functions. For the first part, we use enriched categories. For the second part we use a strong commutative monad, following (Moggi, 1991). For the third part, we define the closure of the product in the base category and in the Kleisli category.

2.2.1. *Module.*

Definition 2.9. An $\mathcal{A}$-module $(M, +, 0, \cdot)$ is an abelian group $(M, +, 0)$ and an operation $(\cdot) : \mathcal{A} \times M \to M$ such that for all a, b in $\mathcal{A}$ and for all x, y in M,

$$a \cdot (x + y) = a \cdot x + a \cdot y, \qquad a \cdot (b \cdot x) = (ab) \cdot x,$$
$$(a + b) \cdot x = a \cdot x + b \cdot x, \qquad 1 \cdot x = x.$$

Lemma 2.10. Let $(M, +, 0, \cdot)$ be an $\mathcal{A}$-module. If $x \in M$ and $a \in \mathcal{A}$,

1 $a \cdot 0 = 0$;
2 $0 \cdot x = 0$;
3 $-(a \cdot x) = a \cdot (-x)$.

Proof. (1) $a \cdot 0 = a \cdot (0 + 0) = a \cdot 0 + a \cdot 0$. By adding the inverse of $a \cdot 0$ to both side of the equality, we get $0 = a \cdot 0$. *(2)* $0 \cdot x = 0 \cdot (2 \cdot x) = 0 \cdot (1 \cdot x + 1 \cdot x) = 0 \cdot x + 0 \cdot x$. Using the same remark as in the previous case, we get that $0 = 0 \cdot x$. *(3)* $a \cdot (-x) + a \cdot x = a \cdot (-x + x) = a \cdot 0 = 0$. Thus $a \cdot (-x)$ is the inverse of $a \cdot x$. □

2.2.2. *Categorical notions.*

These definitions are taken from (Mac Lane, 1998), (Moggi, 1991) and (Kelly, 1982).

Definition 2.11. An object $\top$ in a category $\mathcal{C}$ is called a *terminal object* if for each object A there exists a unique map $\bigcirc_A : A \to \top$.

Definition 2.12. Given a category $\mathcal{C}$ and two objects A and B, if it exists, the *product* of A and B is the data consisting of an object $A \times B$ and two maps $\pi_1^{A,B} : A \times B \to A$ and

$\pi_2^{A,B} : A \times B \to B$, such that for all maps $f : C \to A$ and $g : C \to B$ there exists a unique map $\langle\, f, g\,\rangle : C \to A \times B$ with the following equations holding for all $h : C \to A \times B$:

$$\langle\, f, g\,\rangle; \pi_1^{A,B} = f \tag{41}$$

$$\langle\, f, g\,\rangle; \pi_2^{A,B} = g \tag{42}$$

$$\langle\, h; \pi_1^{A,B}, h; \pi_2^{A,B}\,\rangle = h \tag{43}$$

We usually omit the subscripts A and B in $\pi_i^{A,B}$ when the context is clear.

We say that the category $\mathcal{C}$ *has binary products* if there is a product for all A and B. It is *cartesian* if it has a terminal object and binary products.

Definition 2.13. A *monad* over a category $\mathcal{C}$ is a triple (M, η, μ) where $M : \mathcal{C} \to \mathcal{C}$ is a functor, $\eta : id \dot{\to} M$ and $\mu : M^2 \dot{\to} M$ are natural transformations and the following diagrams commute:

$$\begin{tikzcd}
M^3A \arrow[r, "M\mu_A"] \arrow[d, "\mu_{MA}"'] & M^2A \arrow[d, "\mu_A"] \\
M^2A \arrow[r, "\mu_A"] & MA,
\end{tikzcd} \tag{44}$$

$$\begin{tikzcd}
MA \arrow[r, "\eta_{MA}"] \arrow[dr, "id_{MA}"'] & M^2A \arrow[d, "\mu_A"] & MA. \arrow[l, "M\eta_A"'] \arrow[dl, "id_{MA}"] \\
& MA &
\end{tikzcd} \tag{45}$$

The natural transformation μ is called the *multiplication* of the monad and η the *unit* of the monad.

Definition 2.14. A *strong monad* over a cartesian category $\mathcal{C}$ is a monad (M, η, μ) together with a natural transformation $t_{A,B} : A \times MB \to M(A \times B)$, called the *strength*, such that the diagrams

$$\begin{tikzcd}
\top \times MA \arrow[r, "\simeq"] \arrow[dr, "t"] & MA \\
& M(\top \times A), \arrow[u, "\simeq"']
\end{tikzcd} \tag{46}$$

$$\begin{tikzcd}
(A \times B) \times MC \arrow[d, "t"] & A \times (B \times MC) \arrow[l, "\simeq"'] \arrow[d, "id \times t"] \\
M((A \times B) \times C) & A \times M(B \times C) \arrow[d, "t"] \\
& M(A \times (B \times C)), \arrow[ul, "\simeq"]
\end{tikzcd} \tag{48}$$

$$\begin{tikzcd}
A \times B \arrow[d, "id \times \eta"'] \arrow[dr, "\eta"] & & \\
A \times MB \arrow[r, "t"] & M(A \times B) & \\
A \times M^2B \arrow[u, "id \times \mu"] \arrow[r, "t"] & M(A \times MB) \arrow[r, "Mt"] & M^2(A \times B) \arrow[ul, "\mu"']
\end{tikzcd} \tag{47}$$

commute. The strength is *commutative* if moreover

$$\begin{tikzcd}
MA \times MB \arrow[r, "t"] \arrow[d, "\simeq"'] & M(MA \times B) \arrow[r, "\simeq"] & M(B \times MA) \arrow[r, "t"] & M^2(B \times A) \arrow[d, "\simeq"'] \\
MB \times MA \arrow[r, "t"] & M(MB \times A) \arrow[r, "\simeq"] & M(A \times MB) \arrow[r, "t"] & M^2(A \times B)
\end{tikzcd} \tag{49}$$

commutes. In this case, we write $s_{A,B}$ for the map $MA \times MB \to M(A \times B)$ defined by

$$MA \times MB \xrightarrow{t} M(MA \times B) = M(B \times MA) \xrightarrow{t} M^2(B \times A) \xrightarrow{\mu} M(B \times A) = M(A \times B).$$

By abuse of notation, we refer to s as the strength of the monad.

Lemma 2.15. If M is a commutative, strong monad over a cartesian category $(\mathcal{C}, \times, \top)$, the cartesian structure of $\mathcal{C}$ induces a monoidal structure on $\mathcal{C}_M$. □

Notation 2.16. If $f : A \to MB$ and $g : B \to MC$ are two morphisms of $\mathcal{C}$, we write $f;_M g$ for the map $f; Mg; \mu_C$, that is, for the composition in the Kleisli category. We write $\times_M$ for the tensor of $\mathcal{C}_M$ induced by the product of $\mathcal{C}$, and $\langle\, f, g \,\rangle_M$ for the map $\langle\, f, g \,\rangle; s$.

Lemma 2.17. Using the notations of Definition 2.14, the following equations hold:

$$(50)\quad \begin{tikzcd} & MA \times MB \arrow[d, "s_{A,B}"] \\ A \times B \arrow[ur, "\langle\, \eta_A, \eta_B \,\rangle"] \arrow[r, "\eta_{A\times B}"'] & M(A \times B) \end{tikzcd} \qquad \begin{tikzcd} & MA \times M\top \arrow[d, "s_{\top,A}"] \\ MA \arrow[ur, "\langle\, \eta_\top, id \,\rangle"] \arrow[r, "id"'] & MA \end{tikzcd}\quad (51)$$

Proof. The proof uses extensively the equations of Definition 2.14. □

Definition 2.18. Suppose that $(\mathcal{C}, \otimes, \multimap, \top)$ is a symmetric monoidal closed category. $\mathcal{C}$ is *enriched* over $\mathcal{A}$-modules if

— for any objects X, Y the set $\mathcal{C}(X, Y)$ is equipped with a structure of $\mathcal{A}$-module;
— the composition $\mathcal{C}(X, Y) \times \mathcal{C}(Y, Z) \to \mathcal{C}(X, Z)$ is a bilinear map of $\mathcal{A}$-modules;
— as a mapping of morphisms, the tensor $\otimes : \mathcal{C}(X, Y) \times \mathcal{C}(X', Y') \to \mathcal{C}(X \otimes X', Y \otimes Y')$ is a bilinear map of $\mathcal{A}$-modules;
— the natural isomorphism $\mathcal{C}(X \otimes Y, Z) \to \mathcal{C}(X, Y \multimap Z)$ is a linear map of $\mathcal{A}$-modules.

The unit of the module $\mathcal{C}(X, Y)$ is written $\mathbf{0}$.

2.2.3. *Enriched computational category.* We are now ready to define the category that serves as a basis for interpreting the algebraic computational lambda-calculus.

Definition 2.19. We define an *$\mathcal{A}$-enriched computational category* to be a cartesian category $(\mathcal{C}, \times, \top)$, together with a strong commutative monad (M, η, μ, t), such that

— The Kleisli category $\mathcal{C}_M$ is enriched over $\mathcal{A}$-modules.
— There exists a bifunctor $\Rightarrow : \mathcal{C} \times \mathcal{C}_M \to \mathcal{C}$ such that there is a natural isomorphism $\Psi_\Rightarrow : \mathcal{C}_M(X \times Y, Z) \to \mathcal{C}(X, Y \Rightarrow Z)$. That is, for all maps $f : X \to X'$, $g : Y \to Y'$, $h : Z' \to MZ$ and $k : X' \times Y' \to MZ$ of $\mathcal{C}$, the following equation hold:

$$\Psi_\Rightarrow(f \times g; k; Mh; \mu) = f; \Psi_\Rightarrow(k); (g \Rightarrow h).$$

— The Kleisli category is monoidal closed: there exists a bifunctor $\to$ on $\mathcal{C}_M$ and a natural module isomorphism $\Psi_\to : \mathcal{C}_M(X \times Y, Z) \xrightarrow{\simeq} \mathcal{C}_M(X, Y \to Z)$. That is, for all maps $f : X \to MX'$, $g : Y \to MY'$, $h : Z' \to MZ$ and $k : X' \times Y' \to MZ$ of $\mathcal{C}$, the following equation hold:

$$\Psi_\to(f \times_M g;_M k;_M h) = f;_M \Psi_\to(k);_M (g \to h).$$

Using Notation 2.16, we can rewrite it as follows:

$$\Psi_{\rightarrow}(f \times g; s; Mk; \mu; Mh; \mu) = f; M\Psi_{\rightarrow}(k); \mu; M(g \rightarrow h); \mu.$$

We use the notations $\varepsilon_{X,X}$ for the map $(A \Rightarrow B) \times A \rightarrow MB$ in $\mathcal{C}$ defined by $\Psi_{\Rightarrow}^{-1}(id_{A\Rightarrow B})$ and $\overline{\varepsilon}_{A,B}$ for the map $(A \rightarrow B) \times A \rightarrow MB$ in $\mathcal{C}$ defined by $\Psi_{\rightarrow}^{-1}(\eta_{A\rightarrow B})$.

Lemma 2.20. Suppose that $f : A \rightarrow MA'$, $g : B \rightarrow MB'$ and $h : C \rightarrow MC'$ are three maps of the $\mathcal{A}$-enriched computational category of Definition 2.19. Then the following equations hold.

$$\begin{aligned} \langle\, f + g, h \,\rangle_M &= \langle\, f, h \,\rangle_M + \langle\, g, h \,\rangle_M, & \langle\, h, f + g \,\rangle_M &= \langle\, h, f \,\rangle_M + \langle\, h, g \,\rangle_M, \\ \langle\, \alpha \cdot f, h \,\rangle_M &= \alpha \cdot \langle\, f, h \,\rangle_M, & \langle\, h, \alpha \cdot f \,\rangle_M &= \alpha \cdot \langle\, h, f \,\rangle_M, \\ \langle\, f, \mathbf{0} \,\rangle &= \mathbf{0}, & \langle\, \mathbf{0}, f \,\rangle &= \mathbf{0}. \end{aligned}$$

Proof. Using the fact that the composition of maps and that the tensor induced by the product are bilinear. The first equation is proved as follows.

$$\begin{aligned} \langle\, f + g, h \,\rangle_M &= \langle\, \eta, \eta \,\rangle_M ;_M (f + g \times_M h) \quad \text{by simple rewriting,} \\ &= \langle\, \eta, \eta \,\rangle_M ;_M ((f \times_M h) + (g \times_M h) \quad \text{by bilinearity of } \times_M, \\ &= (\langle\, \eta, \eta \,\rangle_M ;_M (f \times_M h)) + (\langle\, \eta, \eta \,\rangle_M ;_M (g \times_M h)) \end{aligned}$$

by bilinearity of composition in $\mathcal{C}_M$,

$$= \langle\, f, h \,\rangle_M + \langle\, g, h \,\rangle_M \quad \text{by simple rewriting.}$$

The other ones are treated similarly. □

Definition 2.21. Given a map $f : A \rightarrow MB$ in $\mathcal{C}$, we say that f is *base-like* if it satisfies the following properties:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\bigcirc_A"'] & MB \arrow[d, "M\bigcirc_B"] \\ \top \arrow[r, "\eta_\top"] & M\top, \end{tikzcd} \qquad \begin{tikzcd} A \arrow[rr, "f"] \arrow[d, "\langle\, id, id \,\rangle"'] & & MB \arrow[d, "M\langle\, id, id \,\rangle"] \\ A \times A \arrow[r, "f \times f"] & MB \times MB \arrow[r, "s"] & M(B \times B). \end{tikzcd}$$

Lemma 2.22. If $f : A \rightarrow B$ is a map in $\mathcal{C}$, then $f; \eta_B$ is a base-like map. □

2.3. *Concrete models*

In this section, we prove that the categorical setting we just defined describes a consistent structure by providing several concrete models.

We give three models. First, a trivial model: any cartesian category can be made into a model by adding a trivial monad to it. Then, a model based on vector spaces, where both internal homomorphisms are equal. We then show that the category of sets and functions provides a finer model, where the two internal hom are distinct.

2.3.1. *Trivial model.* Consider any cartesian category $(\mathcal{C}, \times, \top)$. The functor sending every object to $\top$ and every map to $id_\top$ is a commutative, strong monad:

— It is trivially a functor.
— The unit of the monad is the unique natural transformation sending any object to $\top$.
— The monad multiplication is $id_\top$.
— The strength is also $id_\top$, remembering that $\top \times \top = \top$.
— The monadic equations are trivially satisfied.

The set of morphisms $\mathcal{C}(X, MY)$ is a module on $\mathcal{A}$ since it contains only one map: consider this map to be the zero of the one-element module.

For any object X and Y, define $X \Rightarrow Y$ and $X \rightarrow Y$ to be $\top$. There are only one possibility for the set-maps $\Psi_\Rightarrow$ and $\Psi_\rightarrow$ since their domains and codomains contain only one map, and these set-maps automatically satisfy the required properties.

2.3.2. *Vector space-based model.* A cartesian closed category that is already enriched over $\mathcal{A}$-modules is an $\mathcal{A}$-enriched computational category, by choosing the identity monad, and setting both functors $\Rightarrow$ and $\rightarrow$ to be the internal hom of the category.

For example, the category of finiteness spaces on some field $\mathcal{K}$ (Ehrhard, 2005) forms an $\mathcal{K}$-enriched computational model: Objects are vector spaces and morphisms are linear maps. The category is therefore enriched over $\mathcal{K}$ vector-spaces (that is, $\mathcal{K}$-modules). The category is a model for linear logic: it is thus cartesian closed and therefore it is a model for the algebraic computational lambda-calculus.

2.3.3. *Set-based model.* The cartesian closed category **Set** of sets and functions provides a model where the functors $\Rightarrow$ and $\rightarrow$ are distinct.

— It is a cartesian closed category.
— Choose M to be the monad arising from the monoidal adjunction between the category **Set** together with its cartesian structure and the category of $\mathcal{A}$-modules and linear maps, with its usual monoidal structure: M is the functor sending a set X to the set of maps $X \rightarrow \mathcal{A}$. Since it comes from a monoidal adjunction, it is a commutative, strong monad.
— If X and Y are any two sets, define $X \Rightarrow Y$ as $\mathbf{Set}(X, MY)$ and $X \rightarrow Y$ as $X \times Y$.
— The map $\Psi_\Rightarrow$ is defined using the fact that the category is closed: it sends $f \in \mathbf{Set}(X \times Y, MZ)$ to the map $x \mapsto (y \mapsto f(x, y))$, element of $\mathbf{Set}(X, \mathbf{Set}(Y, MZ))$.
— The map $\Psi_\rightarrow$ sends the map $f \in \mathbf{Set}(X \times Y, MZ)$ to the map sending x to $(y, z) \mapsto f(x, y)(z)$, which is an element of the set $M(Y \rightarrow Z)$.

2.4. *Denotational semantics*

An $\mathcal{A}$-enriched computational category has the needed structure to serve as a model for the language. In this section, we show that it is possible to encode the language in such a category, and prove that the encoding is sound: if two terms are axiomatically equivalent, then their denotation is equal. We then show that it is possible to construct a syntactic category of base terms that can be equipped with the structure of $\mathcal{A}$-enriched computational category. Finally, we prove that the interpretation of a term in this syntactic category gives back the term, effectively showing completeness.

Table 6. *Interpretation of base terms.*

$$\begin{aligned} \llbracket \Delta, x : A \vdash x : A \rrbracket^v_\Theta &= \llbracket \Delta \rrbracket \times \llbracket A \rrbracket \xrightarrow{\pi_2} \llbracket A \rrbracket \\ \llbracket \Delta \vdash * : \top \rrbracket^v_\Theta &= \llbracket \Delta \rrbracket \xrightarrow{\bigcirc} \top \end{aligned}$$

$$\frac{\llbracket \Delta, x : A \vdash s : B \rrbracket^c_\Theta = \llbracket \Delta \rrbracket \times \llbracket A \rrbracket \xrightarrow{f} M\llbracket B \rrbracket}{\llbracket \Delta \vdash \lambda x.s : A \Rightarrow B \rrbracket^v_\Theta = \llbracket \Delta \rrbracket \xrightarrow{\Psi_{\Rightarrow}(f)} \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket} \quad (52)$$

$$\frac{\llbracket \Delta \vdash u : A_1 \times A_2 \rrbracket^v_\Theta = \llbracket \Delta \rrbracket \xrightarrow{f} \llbracket A_1 \rrbracket \times \llbracket A_2 \rrbracket}{\llbracket \Delta \vdash \pi_i(u) : A_i \rrbracket^v_\Theta = \llbracket \Delta \rrbracket \xrightarrow{f;\pi_i} \llbracket A_i \rrbracket} \quad (53)$$

$$\frac{\begin{aligned} \llbracket \Delta \vdash u : A \rrbracket^v_\Theta &= \llbracket \Delta \rrbracket \xrightarrow{f} \llbracket A \rrbracket \\ \llbracket \Delta \vdash v : B \rrbracket^v_\Theta &= \llbracket \Delta \rrbracket \xrightarrow{g} \llbracket B \rrbracket \end{aligned}}{\llbracket \Delta \vdash \langle u, v \rangle : A \times B \rrbracket^v_\Theta = \llbracket \Delta \rrbracket \xrightarrow{\langle f, g \rangle} \llbracket A \rrbracket \times \llbracket B \rrbracket} \quad (54)$$

Definition 2.23. Consider an $\mathcal{A}$-enriched computational category $\mathcal{C}$. An *evaluation mapping* Φ sends base types to objects of $\mathcal{C}$. Given such an evaluation mapping, we define the denotation of a type $\llbracket A \rrbracket_\Phi$ as follows.

$$\begin{aligned} \llbracket \iota \rrbracket_\Phi &= \Phi(\iota), & \llbracket A \times B \rrbracket_\Phi &= \llbracket A \rrbracket_\Phi \times \llbracket B \rrbracket_\Phi, \\ \llbracket \top \rrbracket_\Phi &= \top, & \llbracket A \Rightarrow B \rrbracket_\Phi &= \llbracket A \rrbracket_\Phi \Rightarrow \llbracket B \rrbracket_\Phi, \\ & & \llbracket A \to B \rrbracket_\Phi &= \llbracket A \rrbracket_\Phi \to \llbracket B \rrbracket_\Phi. \end{aligned}$$

When the context is clear, we omit the subscript Φ.

Definition 2.24. The denotation of a computation is a morphism

$$\llbracket x_1 : A_1, \dots, x_n : A_n \vdash t : B \rrbracket^c_\Phi \quad : \quad \llbracket A_1 \rrbracket_\Phi \times \cdots \times \llbracket A_n \rrbracket_\Phi \longrightarrow M\llbracket B \rrbracket_\Phi$$

of $\mathcal{C}$, called a *c-denotation*, and the denotation of a base term is a morphism

$$\llbracket x_1 : A_1, \dots, x_n : A_n \vdash v : B \rrbracket^v_\Phi \quad : \quad \llbracket A_1 \rrbracket_\Phi \times \cdots \times \llbracket A_n \rrbracket_\Phi \longrightarrow \llbracket B \rrbracket_\Phi$$

of $\mathcal{C}$, called a *b-denotation.* They are defined inductively in Tables 6 and 7. In the definition of b-denotations, u and v are assumed to be base terms, and the s in (52) is any term. In the definition of c-denotations, the following conventions are assumed: in Rule (55), u is a base term; in Rules (59) and (60), the term at the root is assumed to *not* be a base term (because that case would be taken care of by Rule (55)). In particular, neither s nor t are base terms in these rules.

Convention 2.25. When the context is clear, for legibility we use A in place of $\llbracket A \rrbracket$.

The b-denotations and c-denotations are uniquely defined, as proven in the following lemma.

Lemma 2.26. C-denotations and b-denotations of terms are uniquely defined. In particular, Rules (59) and (60) are valid without any restriction on the term at the root of the rule.

Table 7. *Interpretation of computations.*

$$\frac{[\![\Delta \vdash u : A]\!]^v_\Theta = [\![\Delta]\!] \xrightarrow{f} [\![A]\!]}{[\![\Delta \vdash u : A]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{f;\eta_A} M[\![A]\!]} \quad (55)$$

$$\frac{[\![\Delta, x : A \vdash s : B]\!]^c_\Theta = [\![\Delta]\!] \times [\![A]\!] \xrightarrow{f} M[\![B]\!]}{[\![\Delta \vdash \lambda x.s : A \to B]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{\Psi_\to(f)} M([\![A]\!] \to [\![B]\!])} \quad (56)$$

$$\frac{\begin{array}{rl}[\![\Delta \vdash s : A \Rightarrow B]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{f} M([\![A]\!] \Rightarrow [\![B]\!]) \\ [\![\Delta \vdash t : A]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{g} M[\![A]\!]\end{array}}{[\![\Delta \vdash st : B]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{\langle f,g \rangle} M([\![A]\!] \Rightarrow [\![B]\!]) \times M[\![A]\!] \xrightarrow{s;M\varepsilon;\mu} M[\![B]\!]} \quad (57)$$

$$\frac{\begin{array}{rl}[\![\Delta \vdash s : A \to B]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{f} M([\![A]\!] \to [\![B]\!]) \\ [\![\Delta \vdash t : A]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{g} M[\![A]\!]\end{array}}{[\![\Delta \vdash st : B]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{\langle f,g \rangle} M([\![A]\!] \to [\![B]\!]) \times M[\![A]\!] \xrightarrow{s;M\bar{\varepsilon};\mu} M[\![B]\!]} \quad (58)$$

$$\frac{[\![\Delta \vdash s : A_1 \times A_2]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{f} M([\![A_1]\!] \times [\![A_2]\!])}{[\![\Delta \vdash \pi_i(s) : A_i]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{f;M\pi_i} M[\![A_i]\!]} \quad (59)$$

$$\frac{\begin{array}{rl}[\![\Delta \vdash s : A]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{f} M[\![A]\!] \\ [\![\Delta \vdash t : B]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{g} M[\![B]\!]\end{array}}{[\![\Delta \vdash \langle s,t \rangle : A \times B]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{\langle f,g \rangle;s} M([\![A]\!] \times [\![B]\!])} \quad (60)$$

$$\frac{\begin{array}{rl}[\![\Delta \vdash s : A]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{f} M[\![A]\!] \\ [\![\Delta \vdash t : A]\!]^c_\Theta & = [\![\Delta]\!] \xrightarrow{g} M[\![A]\!]\end{array}}{[\![\Delta \vdash s + t : A]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{f+g} M[\![A]\!]} \quad (61)$$

$$\frac{[\![\Delta \vdash s : A]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{f} M[\![A]\!]}{[\![\Delta \vdash \alpha \cdot s : A]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{\alpha \cdot f} M[\![A]\!]} \quad (62) \qquad \frac{}{[\![\Delta \vdash \mathbf{0} : A]\!]^c_\Theta = [\![\Delta]\!] \xrightarrow{\mathbf{0}} M[\![A]\!]} \quad (63)$$

Proof. We prove the lemma by induction on the size of a term. We consider each rule separately and show they are valid. The non-base-term cases being the original rules, we only have to consider the base-terms.

Rule (59). The term s is a base term: the typing derivation is of the form $\Delta \vdash \pi_i(u) : A_i$.

The c-denotation is either defined by Rule (55) as follows:

$$\frac{\dfrac{[\![\Delta \vdash u : A_1 \times A_2]\!]^v = [\![\Delta]\!] \xrightarrow{f} [\![A_1]\!] \times [\![A_2]\!]}{[\![\Delta \vdash \pi_i(u) : A_i]\!]^v = [\![\Delta]\!] \xrightarrow{f;\pi_i} [\![A_i]\!]}}{[\![\Delta \vdash \pi_i(u) : A_i]\!]^c = [\![\Delta]\!] \xrightarrow{f;\pi_i} [\![A_i]\!] \xrightarrow{\eta_{A_i}} M[\![A_i]\!]}$$

or directly by Rule (59), yielding the map

$$\frac{[\![\Delta \vdash u : A_1 \times A_2]\!]^c = [\![\Delta]\!] \xrightarrow{f'} M([\![A_1]\!] \times [\![A_2]\!])}{[\![\Delta \vdash \pi_i(u) : A_i]\!]^c = [\![\Delta]\!] \xrightarrow{f';M\pi_i} M[\![A_i]\!]}$$

By induction hypothesis, since u is a base term, its c-denotation comes from Rule (55) and $f' = f; \eta_{A_1 \times A_2}$. We can conclude that the two denotations of $\pi_i(u)$ are equal by naturality of η.

Rule (60). The terms s and t are base terms: we are in fact considering a typing derivation of the form $\Delta \vdash \langle\, u, v \,\rangle : A \times B$. Note that the b-denotation is uniquely defined. The c-denotation is either defined by Rule (55) as follows

$$\dfrac{\dfrac{[\![\, \Delta \vdash u : A \,]\!]^v = [\![\, \Delta \,]\!] \xrightarrow{f} [\![\, A \,]\!] \quad [\![\, \Delta \vdash v : B \,]\!]^v = [\![\, \Delta \,]\!] \xrightarrow{g} [\![\, B \,]\!]}{[\![\, \Delta \vdash \langle\, u, v \,\rangle : A \times B \,]\!]^v = [\![\, \Delta \,]\!] \xrightarrow{\langle\, f,g \,\rangle} [\![\, A \,]\!] \times [\![\, B \,]\!]}}{[\![\, \Delta \vdash \langle\, u, v \,\rangle : A \times B \,]\!]^c = [\![\, \Delta \,]\!] \xrightarrow{\langle\, f,g \,\rangle; \eta_{A \times B}} M([\![\, A \,]\!] \times [\![\, B \,]\!])}$$

or by Rule (60)

$$\dfrac{[\![\, \Delta \vdash u : A \,]\!]^c = [\![\, \Delta \,]\!] \xrightarrow{f'} M[\![\, A \,]\!] \quad [\![\, \Delta \vdash v : B \,]\!]^c = [\![\, \Delta \,]\!] \xrightarrow{g'} M[\![\, B \,]\!]}{[\![\, \Delta \vdash \langle\, u, v \,\rangle : A \times B \,]\!]^c = [\![\, \Delta \,]\!] \xrightarrow{\langle\, f',g' \,\rangle} M[\![\, A \,]\!] \times M[\![\, B \,]\!] \xrightarrow{s} M([\![\, A \,]\!] \times [\![\, B \,]\!]).}$$

By induction hypothesis, since u and v are base terms we have $f' = f; \eta_A$ and $g' = g; \eta_B$. Using Rule (50) of Lemma 2.17, we can conclude that the two maps are equal.

This closes the induction. $\square$

2.4.1. *Soundness.* We first show that if two terms are axiomatically equivalent, their denotations are equal.

Lemma 2.27. Suppose that u is a base term, that $\Delta \vdash u : A$ has f for c-denotation and that $\Delta, x : A \vdash s : B$ has g for c-denotation. Then $h = [\![\, \Delta \vdash s[x \leftarrow u] : B \,]\!]^c$ is equal to the map

$$[\![\, \Delta \,]\!] \xrightarrow{\langle\, \eta,f \,\rangle} M[\![\, \Delta \,]\!] \times M[\![\, A \,]\!] \xrightarrow{s} M([\![\, \Delta \,]\!] \times [\![\, A \,]\!]) \xrightarrow{Mg;\mu} M[\![\, B \,]\!].$$

Now, suppose that s and u are base terms and have respectively f' and g' for b-denotation. Then the morphism $[\![\, \Delta \vdash s[x \leftarrow u] : B \,]\!]^v$ is equal to

$$[\![\, \Delta \,]\!] \xrightarrow{\langle\, id,f' \,\rangle} [\![\, \Delta \,]\!] \times [\![\, A \,]\!] \xrightarrow{g'} [\![\, B \,]\!].$$

Proof. The proof is done by structural induction on the typing derivation $\Delta, x : A \vdash s : B$.

$\Delta, x : A \vdash * : \top$. In this case, $g = h = \bigcirc; \eta_\top$. The required equation is shown as follows:

$$\begin{tikzcd}
{[\![\, \Delta \,]\!]} \arrow[r, "\langle\, \eta,f \,\rangle"] \arrow[dd, "\bigcirc"'] \arrow[dr, phantom, "(a)"] & M[\![\, \Delta \,]\!] \times M[\![\, A \,]\!] \arrow[r, "s"] \arrow[d, "M\bigcirc \times M\bigcirc"] & M([\![\, \Delta \,]\!] \times [\![\, A \,]\!]) \arrow[r, "M\bigcirc"] \arrow[ddr, "M\bigcirc"'] \arrow[d, phantom, "(c)"] \arrow[dr, phantom, "(d)"] & M\top \arrow[r, "M\eta_\top"] \arrow[dd, "id"] \arrow[dr, phantom, "(e)"] & MM\top \arrow[ddl, bend left, "\mu_\top"] \\
 & M\top \times M\top \arrow[drr, "s"] \arrow[d, phantom, "(b)"] & & & \\
\top \arrow[ur, "\eta_\top \times \eta_\top"] \arrow[rrr, "\eta_\top"'] & & & M\top &
\end{tikzcd}$$

Square (a) commutes respectively due to the naturality of η and due to f being base-like, using Lemma 2.22. Triangle (b) commutes because of Eq. (50). Square (c) commutes by naturality of s. Square (d) commutes trivially. Triangle (e) commutes because of Eq. (45).

$\Delta, x : A \vdash x : A$. In this case, $g = \pi_2; \eta_A$ and $h = f$. We can decompose the desired equation as follows:

$$\begin{tikzcd}
\llbracket \Delta \rrbracket \arrow[r, "\langle \eta, f \rangle"] \arrow[dd, "f"'] \arrow[dr, phantom, "(a)"] & M\llbracket \Delta \rrbracket \times M\llbracket A \rrbracket \arrow[r, "s"] \arrow[d, "M\bigcirc \times Mid"] \arrow[dr, phantom, "(c)"] & M(\llbracket \Delta \rrbracket \times \llbracket A \rrbracket) \arrow[r, "M\pi_2"] \arrow[ddr, "M\pi_2"'] \arrow[dr, phantom, "(d)"] & M\llbracket A \rrbracket \arrow[r, "M\eta_A"] \arrow[dd, "id"] \arrow[dr, phantom, "(e)"] & MM\llbracket A \rrbracket \arrow[ddl, bend left, "\mu_A"] \\
 & M\top \times M\llbracket A \rrbracket \arrow[ddrr, "s"] \arrow[d, phantom, "(b)"] & & & \\
M\llbracket A \rrbracket \arrow[ur, "\eta_\top \times id"'] \arrow[rrr, "id"'] & & & M\llbracket A \rrbracket &
\end{tikzcd}$$

Square (a) commutes due to the naturality of η (and remembering that $\bigcirc_\top = id_\top$) Triangle (b) commutes because of Eq. (51). Square (c) commutes by naturality of s (and by remembering that $\pi_2 = \bigcirc \times id$). Square (d) commutes trivially. Triangle (e) commutes because of Eq. (45).

$\Delta, x : A \vdash \lambda y.r : C \to D$. By uniqueness of the typing derivation (Lemma 2.2), we have $\Delta, x : A, y : C \vdash r : D$. Let its c-denotation be k. By induction hypothesis (and by Lemma 2.3), if the denotation $\llbracket \Delta, y : C \vdash r[x \leftarrow u] : D \rrbracket^c$ is written k' then

$$k' = \llbracket \Delta \rrbracket \times \llbracket C \rrbracket \xrightarrow{\langle \eta, f \rangle \times \eta} M\llbracket \Delta \rrbracket \times M\llbracket A \rrbracket \times M\llbracket C \rrbracket$$
$$\xrightarrow{s} M(\llbracket \Delta \rrbracket \times \llbracket A \rrbracket \times \llbracket C \rrbracket) \xrightarrow{Mk;\mu} M\llbracket D \rrbracket.$$

Since $s[x \leftarrow u] = \lambda y.r[x \leftarrow u]$, the map h is equal to $\Psi_\to(k')$. On the other hand, we have $g = \Psi_\to(k)$. By naturality of $\Psi_\to$, the map $\Psi_\to(k')$ is equal to

$$\langle \eta, f \rangle; s; M\Psi_\to(k); \mu; M(\eta \to \eta); \mu,$$

which is the requested map since $\eta_C \to \eta_D = \eta_{C \to D}$.

$\Delta, x : A \vdash r_1 r_2 : B$ *when* $r_1 : C \to D$. In this case, the typing derivations $\Delta, x : A \vdash r_1 : C \to B$ and $\Delta, x : A \vdash r_2 : C$ are valid. Let their c-denotations be respectively k_1 and k_2. By induction hypothesis, if we write $\llbracket \Delta \vdash r_1[x \leftarrow u] : C \to B \rrbracket^c$ as k_1' and $\llbracket \Delta \vdash r_1[x \leftarrow u] : C \rrbracket^c$ as k_2', then

$$k_1' = \llbracket \Delta \rrbracket \xrightarrow{\langle \eta, f \rangle} M\llbracket \Delta \rrbracket \times M\llbracket A \rrbracket \xrightarrow{s} M(\llbracket \Delta \rrbracket \times \llbracket A \rrbracket) \xrightarrow{Mk_1;\mu} M(\llbracket C \rrbracket \to \llbracket B \rrbracket),$$
$$k_2' = \llbracket \Delta \rrbracket \xrightarrow{\langle \eta, f \rangle} M\llbracket \Delta \rrbracket \times M\llbracket A \rrbracket \xrightarrow{s} M(\llbracket \Delta \rrbracket \times \llbracket A \rrbracket) \xrightarrow{Mk_2;\mu} M\llbracket C \rrbracket,$$

and we have $h = \langle k_1', k_2' \rangle; s; M\bar{\varepsilon}; \mu$. To prove the desired equality, we use the fact that f is base-like and Definition 2.21.

The remaining cases are similar. □

Theorem 2.28 (Soundness). Suppose that $\Delta \vdash s \simeq_{ax} t : A$. Then $\llbracket s \rrbracket^c = \llbracket t \rrbracket^c$. Suppose that $\Delta \vdash u \simeq_{ax} v : A$. Then $\llbracket u \rrbracket^v = \llbracket v \rrbracket^v$.

Proof. The proof is done by structural induction on the proof of $s \simeq_{ax} t$.

Algebraic rules of Table 3. Note that these rules do not relate base terms, just computations. The denotation is therefore in the Kleisli category: morphisms in this category forms a module, and the rules of Table 3 precisely states the module equations of Definition 2.9.

Distributive rules of Table 4. Again, note that the rules only related computations, not base terms. The interpretations of the terms in relations live in the Kleisli category. The desired result for these rules comes from the fact that the Kleisli category is enriched as a symmetric monoidal closed category.

Rules (4)*,* (6)*,* (8)*,* (10)*,* (12) *and* (14). From the fact that the set-application $\times : \mathcal{C}_M(X,Y) \times \mathcal{C}_M(X',Y') \to \mathcal{C}_M(X \times X', Y \times Y')$ is a bilinear module homomorphism.

Rules (16)*,* (18)*,* (17)*,* (20)*,* (19) *and* (21). Using Lemma 2.20. We show how to deal with Rule (16). If $f = [\![\, \Delta \vdash r : A \,]\!]^c$, $g = [\![\, \Delta \vdash s : A \,]\!]^c$ and $h = [\![\, \Delta \vdash t : B \,]\!]^c$, then

$$\begin{aligned} [\![\, \Delta \vdash \langle\, r+s, t \,\rangle : A \times B \,]\!]^c &= \langle\, f+g, h \,\rangle_M && \text{by definition,} \\ &= \langle\, f, h \,\rangle_M + \langle\, g, h \,\rangle_M && \text{by Lemma 2.20.} \end{aligned}$$

The other cases are similar.

Rules (5)*,* (7)*,* (9)*,* (11)*,* (13) *and* (15). Using the fact that the composition in the Kleisli category is bilinear. Example with Rule (5): provided that f is the denotation $[\![\, \Delta \vdash s : A \times B \,]\!]^c$ and $g = [\![\, \Delta \vdash t : A \times B \,]\!]^c$,

$$\begin{aligned} [\![\, \Delta \vdash \pi_1(s+t) : A \,]\!]^c &= (f+g);_M M(\pi_1) && \text{by definition,} \\ &= f;_M M(\pi_1) + g;_M M(\pi_1) && \text{by bilinearity.} \end{aligned}$$

Rules (22)*,* (23) *and* (24). From the bilinearity of $\Psi_\to$. For example, Rule (22) is proven as follows. Provided that $f = [\![\, \Delta, x : A \vdash s : B \,]\!]^c$ and that g is the map $[\![\, \Delta, x : A \vdash t : B \,]\!]^c$,

$$\begin{aligned} [\![\, \Delta \vdash \lambda x.(s+t) : A \to B \,]\!]^c &= \Psi_\to(f+g) && \text{by definition,} \\ &= \Psi_\to(f) + \Psi_\to(g) && \text{by bilinearity.} \end{aligned}$$

Computational rules of Table 5. Rules (25)*,* (27) *and* (29). Correct from the cartesian structure of $\mathcal{C}$.

Rule (31). If $\Delta \vdash u : \top$ is a base term, its b-denotation is a map $[\![\, \Delta \,]\!] \to \top$ in $\mathcal{C}$. Since $\top$ is a terminal object in $\mathcal{C}$, there is only one such map: It is therefore equal to the b-denotation of $\Delta \vdash * : \top$. For the c-denotation, we fall back to the b-denotation because of Rule (55).

Rule (33). If $\Delta \vdash s : A$ has f for c-denotation, $\Delta \vdash (\lambda x.x)s : A$ has $(\Psi_\to(\eta_A) \times_M f);_M \bar{\varepsilon}_{A,A}$, where $\Psi_\to(\eta_A) : \top \to M(A \to A)$. This can be rewritten as the composition of f with the map

$$(\Psi_\to(\eta_A) \times_M \eta_A);_M \Psi_\to^{-1}(\eta_A).$$

By naturality of $\Psi_\to$, the latter map is the identity and the composition is therefore f.

Rule (35). Since we deal with applications, only c-denotations are defined in this case. Then $\Delta \vdash t : A$ and $\Delta \vdash s : A \to B$. If the former has f for c-denotation and the latter g, then the c-denotation of the left-hand-side of the equation is

$$\langle\, \Psi_{\to}((f \times_M \eta_{A\to B});_M \bar{\varepsilon}_{A,B}), g \,\rangle_M ;_M \bar{\varepsilon}_{A\to B,B} \tag{64}$$

Since $\bar{\varepsilon}_{A,B} = \Psi_{\to}^{-1}(\eta_{A\to B})$, by naturality of $\Psi_{\to}$ we have $(\eta_{A\to B} \times_M f);_M \bar{\varepsilon}_{A,B} = \Psi_{\to}^{-1}(f \to B)$. Therefore (64) can be rewritten as

$$\langle\, (f \to B), g \,\rangle_M ;_M \bar{\varepsilon}.$$

Again by naturality of $\Psi_{\to}$, this can be reformulated as $g; (f \to B)$. But this is exactly $\langle\, f, g \,\rangle_M ;_M \bar{\varepsilon}$, that is, the c-denotation of $\Delta \vdash st : B$.

Rule (26). A judgment $\Delta \vdash (\lambda x.s)v : B$ is always a computation. Provided that the variable x is of type A, write $f = [\![\, \Delta, x : A \vdash s : B \,]\!]^c$ and $g = [\![\, \Delta \vdash v : A \,]\!]^c$. From Rule (55), g is of the form $h; \eta_A$, where $h = [\![\, \Delta \vdash v : A \,]\!]^v$. The morphism $[\![\, \Delta \vdash (\lambda x.s)v : B \,]\!]^c$ is equal to $\langle\, \Psi_{\to}(f), g \,\rangle; s; M\bar{\varepsilon}_{A,B}; \mu$. Since $\bar{\varepsilon}_{A,B}$ is the map $\Psi_{\to}^{-1}(id_{A\to B})$, by naturality it is equal to $\langle\, \eta, h \,\rangle; s; M(f); \mu$. Using Lemma 2.27, this is equal to $[\![\, \Delta \vdash s[x \leftarrow v] : B \,]\!]$.

Rule (28). The denotations of the two sides of the equivalence relation are equal because of the natural isomorphism between $\mathcal{C}(X, M(Y \to Z))$ and $\mathcal{C}(X, Y \Rightarrow Z)$.

Rule (30). For the two sides of the equation to be well-typed, s must be of type $A \to B$ and x of type A must not be free variable of s. Let $f = [\![\, \Delta \vdash s : A \to B \,]\!]^c$. The map $[\![\, \Delta \vdash \lambda x.sx : A \to B \,]\!]^c$ is equal to $\Psi_{\to}((f \times \eta_A); s; \bar{\varepsilon}_{A,B})$. Again, by naturality this is equal to f.

Rule (32). Similar to (30) using the fact that $\Psi_{\Rightarrow}$ is a natural mapping.

Rule (34). Because of the naturality of $\Psi_{\to}$. If y is of type C, g is the denotation of t and f the denotation of s, the left-hand-side has for c-denotation the morphism $\langle\, \eta_\Delta, g \,\rangle_M ;_M \Psi_{\to}(f)$. This is equal to the map $\Psi_{\to}(\langle\, \eta_\Delta, g \,\rangle_M \times_M \eta_C ;_M f)$, which is the c-denotation of the right-hand-side.

Rule (36). This amounts to say that $(f \times_M g);_M ; h = (f \times_M \eta);_M (\eta \times_M g);_M ; h$. It is correct since the Kleisli category is symmetric monoidal.

Rule (37). Again, this is true since the category is monoidal closed.

Rule (38). Because the monad is strongly commutative.

Rule (39) *and* (40). By naturality of $\Psi_{\to}$.

Finally, for the congruence rules, it is sufficient to use the compositionality of the denotation. □

2.4.2. *Completeness.* We want to show that if for any model the denotations of two terms are equal, then they are axiomatically equivalent. As in (Lambek and Scott, 1989), we use the notion of internal language: We define a syntactic category of base terms where the equality on morphisms is the axiomatic equivalence. Then we show that if a term is interpreted in this category, its denotation is axiomatically equivalent to itself.

Notation 2.29. We shall use the following notations.

$$\begin{aligned}
\mathit{let}\ x = s\ \mathit{in}\ t &\equiv t[x \leftarrow s],\\
\mathit{let}\ \langle\, x, y \,\rangle = s\ \mathit{in}\ t &\equiv t[y \leftarrow (\pi_1 s), y \leftarrow (\pi_2 s)],\\
\mathit{let}^{\circ}\ x = s\ \mathit{in}\ t &\equiv (\lambda x.t)s,\\
\Lambda {*}.s &\equiv \Lambda x.s \quad \text{where } x \text{ is a fresh variable of type } \top,\\
[\, s \,] &\equiv \Lambda {*}.s,\\
\{\, s \,\} &\equiv s{*},\\
MA &\equiv (\top \Rightarrow A).
\end{aligned}$$

The notations $[\, - \,]$ and $\{\, - \,\}$ follow the conventions used in (Filinski, 1996) for dealing with monads.

Lemma 2.30. Provided that both sides are well-typed, the following rules are valid.

$$\begin{aligned}
(\Lambda x.s)u &\simeq_{ax} s[x \leftarrow u] && (65)\\
[\, \{\, u \,\} \,] &\simeq_{ax} u && (66)\\
\{\, [\, s \,] \,\} &\simeq_{ax} s && (67)\\
(\lambda x.\{\, s \,\})t &\simeq_{ax} \{\, (\lambda x.s)t \,\} && (68)\\
(\lambda y.(r(\pi_1 y))(\pi_2 y))\langle\, s, t \,\rangle &\simeq_{ax} (rs)t && (69)\\
(\lambda x.\langle\, r, s \,\rangle)t &\simeq_{ax} \langle\, r, (\lambda x.s)t \,\rangle && (70)\\
(\lambda x.\langle\, r, s \,\rangle)t &\simeq_{ax} \langle\, (\lambda x.r)t, s \,\rangle && (71)\\
\{\, s + t \,\} &\simeq_{ax} \{\, s \,\} + \{\, t \,\} && (72)\\
\{\, \alpha \cdot s \,\} &\simeq_{ax} \alpha \cdot \{\, s \,\} && (73)
\end{aligned}$$

Proof. (65): Using (28) and (26). (66): Using (31) and (32). (67): Using (65). (68): By the following equivalences:

$$\begin{aligned}
(\lambda x.\{\, s \,\})t &= (\lambda x.s{*})t\\
&\simeq_{ax} (\lambda y.((\lambda x.s{*})t)) {*} \quad \text{for } y \text{ fresh, by Eq. (26),}\\
&\simeq_{ax} ((\lambda x.\lambda y.s{*})t) {*} \quad \text{by Eq. (34),}\\
&= \{\, (\lambda x.\lambda y.s{*})t \,\}\\
&\simeq_{ax} \{\, (\lambda x.\lambda y.sy)t \,\} \quad \text{by Eq. (31),}\\
&\simeq_{ax} \{\, (\lambda x.s)t \,\} \quad \text{by Eq. (30).}
\end{aligned}$$

(69): By the following equivalences:

$$\begin{aligned}
(\lambda z.(r(\pi_1 z))(\pi_2 z))\langle\, s, t \,\rangle &\simeq_{ax} (\lambda y.(\lambda x.(\lambda z.(r(\pi_1 z))(\pi_2 z))\langle\, x, y \,\rangle)s)t \quad \text{by Eq. (36),}\\
&\simeq_{ax} (\lambda y.(\lambda x.(rx)y)s)t \quad \text{by Eq. (26),(25) and (27),}\\
&\simeq_{ax} (\lambda x.(\lambda y.(rx)y)t)s \quad \text{by Eq. (38),}\\
&\simeq_{ax} (\lambda x.(rx)t)s \quad \text{by Eq. (30),}\\
&\simeq_{ax} (\lambda x.(\lambda z.zt)(rx))s \quad \text{by Eq. (35),}\\
&\simeq_{ax} (\lambda z.zt)((\lambda x.rx)s) \quad \text{by Eq. (37),}\\
&\simeq_{ax} (\lambda z.zt)(rs) \quad \text{by Eq. (30),}
\end{aligned}$$

$(rs)t$ by Eq. (35).

(70): By the following sequence of equivalences:

$$\begin{aligned}(\lambda x.\langle\, r,s\,\rangle)t \simeq_{ax} & (\lambda x.(\lambda z.z)\langle\, r,s\,\rangle)t \quad \text{by Eq. (33)},\\ \simeq_{ax} & (\lambda x.(\lambda y.(\lambda z.(\lambda z.z)\langle\, z,y\,\rangle)r)s)t \quad \text{by Eq. (36)},\\ \simeq_{ax} & (\lambda y.(\lambda z.(\lambda z.z)\langle\, z,y\,\rangle)r)((\lambda x.s)t) \quad \text{by Eq. (37)},\\ \simeq_{ax} & (\lambda z.z)\langle\, r,(\lambda x.s)t\,\rangle \quad \text{by Eq. (36)},\\ \simeq_{ax} & \langle\, r,(\lambda x.s)t\,\rangle \quad \text{by Eq. (33)}.\end{aligned}$$

(71): By the same sequence of equalities, plus two uses of (38). (72): Using (16). (73): Using (17). □

Lemma 2.31. One can define a cartesian category whose objects are types and whose morphisms $A \to B$ are typing judgments $x : A \vdash v : B$ when v is a base term. Equality on morphisms is given by the axiomatic equivalence.

— The identity morphism on A is the typing judgment $x : A \vdash x : A$, and the composition of $x : A \vdash u : B$ and $y : B \vdash v : C$ is $x : A \vdash \mathit{let}\ y = u\ \mathit{in}\ v : C$.
— The product of A and B is $A \times B$; the product of $x : A \vdash u : B$ and $y : C \vdash v : D$ is $z : A \times C \vdash \mathit{let}\ \langle\, x,y\,\rangle = z\ \mathit{in}\ \langle\, u,v\,\rangle : B \times D$; the first and second projections are $x : A \times B \vdash \pi_1(x) : A$ and $x : A \times B \vdash \pi_2(x) : B$. If f is the map $x : C \vdash u : A$ and g is the map $x : C \vdash v : B$, then $\langle\, f,g\,\rangle$ is $x : C \vdash \langle\, u,v\,\rangle : A \times B$.
— The terminal object is $\top$, and the unique map from any object A to $\top$ is the judgment $x : A \vdash * : \top$.

Proof. The composition is well defined by Lemma 2.4. It is associative because the term construct *let* stands for the substitution (Notation 2.29), and the unit is indeed the unit of the composition for the same reasons.

To show that $\times$ is a product, it is enough to show that the three equations of Definition 2.12 hold.

Equation (41). The left hand side is $x : C \vdash \mathit{let}\ y = \langle\, u,v\,\rangle\ \mathit{in}\ \pi_1(y) : A$. It is immediate to see that this is axiomatically equivalent to u using Rule (25).

Equation (42). Similarly, the left-hand-side is $x : C \vdash \mathit{let}\ y = \langle\, u,v\,\rangle\ \mathit{in}\ \pi_2(y) : B$. Using Rule (27), this is immediately axiomatically equivalent to v,

Equation (43). Let $h : C \to A \times B$ be the base term $x : C \vdash u : A \times B$. The left-hand-side of the equation is the typing judgment

$$x : C \vdash \langle\, \pi_1(u), \pi_2(u)\,\rangle.$$

We get the right-hand-side by applying Rule (29).

Finally, $\top$ is a terminal object since any typing judgment $x : A \vdash u : \top$ is axiomatically equivalent to $*$ by Rule (31). □

Definition 2.32. The category described in Lemma 2.31 is called the *category of base terms*, and is denoted with $\mathcal{C}_l$.

Theorem 2.33. The category $\mathcal{C}_l$ admits a commutative, strong monad. The monad M sends A to MA and $x : A \vdash u : B$ to $y : MA \vdash [\ let^\circ\, x = \{\ y\ \}\ in\ u\ ] : MB$, and the three required morphisms are

$$\begin{aligned}
\eta_A &= x : A \vdash [\ x\ ] : MA, \\
\mu_A &= x : MMA \vdash [\ \{\ \{\ x\ \}\ \}\ ] : MA, \\
t_{A,B} &= x : A \times MB \vdash let\ \langle\ y, z\ \rangle = x\ in\ [\ \langle\ y, \{\ z\ \}\ \rangle\ ] : M(A \times B).
\end{aligned}$$

The Kleisli category $\mathcal{C}_{l\,M}$ is enriched over $\mathcal{A}$-modules. The module structure of $\mathcal{C}_l(A, MB)$ is given by the module structure of the term algebra. Consider the two maps $f = (x : A \vdash u : MB)$ and $g = (x : A \vdash v : MB)$. We define

$$\begin{aligned}
0 &= x : A \vdash [\ \mathbf{0}\ ] : MB, \\
f + g &= x : A \vdash [\ \{\ u\ \} + \{\ v\ \}\ ] : MB, \\
\alpha \cdot f &= x : A \vdash [\ \alpha \cdot \{\ u\ \}\ ] : MB.
\end{aligned}$$

Proof. We first show that M is a functor:

— If id_A is the identity on A, the map Mid_A is by definition

$$y : MA \vdash [\ let^\circ\, x = \{\ y\ \}\ in\ x\ ] : MA$$

which is equal to y using Rule (33) and (66) (Lemma 2.30).

— If $f : A \to B$ and $g : B \to C$ are the morphisms $x : A \vdash u : B$ and $y : B \vdash v : C$,

$$\begin{aligned}
Mf &= z_1 : MA \vdash [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ] : MB, \\
Mg &= z_2 : MB \vdash [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ] : MC, \\
M(f;g) &= z_1 : MA \vdash [\ let^\circ\, x = \{\ z_1\ \}\ in\ let^\circ\, y = u\ in\ y\ ] : MC. \qquad (74)
\end{aligned}$$

The map $Mf; Mg$ is

$$\begin{aligned}
z_1 : MA \vdash\ & let\ z_2 = [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ in\ [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ], \\
&\simeq_{ax} [\ let^\circ\, y = \{\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ \}\ in\ v\ ] \quad \text{by (26)}, \\
&\simeq_{ax} [\ let^\circ\, y = (let^\circ\, x = \{\ z_1\ \}\ in\ u)\ in\ v\ ] \quad \text{by (67)}, \\
&\simeq_{ax} [\ let^\circ\, x = \{\ z_1\ \}\ in\ let^\circ\, y = u\ in\ v\ ] : MC, \quad \text{by (37)}
\end{aligned}$$

which is equal to Eq. (74).

The maps η, μ and t are natural transformations. Indeed, if $f : A \to B$ and $g : C \to D$ are the morphisms $x : A \vdash u : B$ and $y : C \vdash v : D$ then

$$\begin{aligned}
Mf &= z_1 : MA \vdash [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ] : MB, \\
M^2 f &= t_1 : M^2 A \vdash [\ let^\circ\, z1 = \{\ t_1\ \}\ in\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ ] : M^2 A, \\
Mg &= z_2 : MC \vdash [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ] : MD, \\
f \times g &= z_3 : A \times C \vdash let\ \langle\ x, y\ \rangle = z_3\ in\ \langle\ u, v\ \rangle : B \times D, \\
f \times Mg &= z_4 : A \times MC \vdash let\ \langle\ x, z_2\ \rangle = z_4\ in\ \langle\ u, [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ]\ \rangle : B \times MD, \\
M(f \times g) &= z_5 : M(A \times C) \vdash \left[\begin{array}{l} let^\circ\, z_3 = \{\ z_5\ \}\ in \\ let\ \langle\ x, y\ \rangle = z_3\ in\ \langle\ u, v\ \rangle \end{array}\right] : M(B \times D).
\end{aligned}$$

In this case, the equations are proven as follows.

— The map $f;\eta_B$ is equal to

$$\begin{aligned} x : A \vdash\, & let\ \ z = u\ in\ [\ z\ ] \\ & = [\ u\ ] : MB \end{aligned}$$

The map $\eta_A; Mf$ is equal to

$$\begin{aligned} x : A \vdash\, & let\ \ z_1 = [\ x\ ]\ in\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ] \\ & = [\ let^\circ\, x = \{\ [\ x\ ]\ \}\ in\ u\ ] \\ & \simeq_{ax} [\ let^\circ\, x = x\ in\ u\ ] \quad \text{by Eq. (67),} \\ & \simeq_{ax} [\ u\ ] : MB \quad \text{by Eq. (26).} \end{aligned}$$

The two maps are equal: η is a natural transformation.

— The map $M^2 f;\mu_B$ is equal to

$$\begin{aligned} t_1 : M^2A \vdash\, & let\ t_2 = [\ let^\circ\, z_1 = \{\ t_1\ \}\ in\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ ]\ in\ [\ \{\ \{\ t_2\ \}\ \}\ ] \\ & = [\ \{\ \{\ [\ let^\circ\, z_1 = \{\ t_1\ \}\ in\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ ]\ \}\ \}\ ] \\ & \simeq_{ax} [\ \{\ let^\circ\, z_1 = \{\ t_1\ \}\ in\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ \}\ ] \quad \text{by Eq. (67),} \\ & \simeq_{ax} [\ let^\circ\, z_1 = \{\ t_1\ \}\ in\ \{\ [\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ]\ \}\ ] \quad \text{by Eq. (68),} \\ & \simeq_{ax} [\ let^\circ\, z_1 = \{\ t_1\ \}\ in\ let^\circ\, x = \{\ z_1\ \}\ in\ u\ ] \quad \text{by Eq. (67),} \\ & \simeq_{ax} [\ let^\circ\, x = (let^\circ\, z1 = \{\ t_1\ \}\ in\ \{\ z_1\ \})\ in\ u\ ] \quad \text{by Eq. (37),} \\ & \simeq_{ax} [\ let^\circ\, x = \{\ let^\circ\, z1 = \{\ t_1\ \}\ in\ z_1\ \}\ in\ u\ ] \quad \text{by Eq. (68),} \\ & \simeq_{ax} [\ let^\circ\, x = \{\ \{\ t_1\ \}\ \}\ in\ u\ ] : MB \quad \text{by Eq. (33).} \end{aligned}$$

The map $\mu_A; Mf$ is equal to

$$\begin{aligned} t_1 : M^2A \vdash\, & let\ t_1 = [\ \{\ \{\ t_1\ \}\ \}\ ]\ in\ [\ let^\circ\, x = \{\ t_1\ \}\ in\ u\ ] \\ & = [\ let^\circ\, x = \{\ [\ \{\ \{\ t_1\ \}\ \}\ ]\ \}\ in\ u\ ] \\ & \simeq_{ax} [\ let^\circ\, x = \{\ \{\ t_1\ \}\ \}\ in\ u\ ] : MB \quad \text{by Eq. (67).} \end{aligned}$$

The two maps are equal: μ is a natural transformation.

— The map $(f \times Mg); t_{B,D}$ is equal to

$$\begin{aligned} z_4 : A \times MC \vdash\, & \left(\begin{array}{l} let\ t = (let\ \langle\ x, z_2\ \rangle = z_4\ in\ \langle\ u\ ,\ [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ]\ \rangle)\ in \\ let\ \langle\ y, z\ \rangle = t\ in\ [\ \langle\ y, \{\ z\ \}\ \rangle\ ] \end{array} \right) \\ & = \left(\begin{array}{l} let\ \langle\ x, z_2\ \rangle = z_4\ in \\ let\ \langle\ y, z\ \rangle = \langle\ u\ ,\ [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ]\ \rangle\ in \\ [\ \langle\ y, \{\ z\ \}\ \rangle\ ] \end{array} \right) \\ & \simeq_{ax} let\ \langle\ x, z_2\ \rangle = z_4\ in\ [\ \langle\ u\ ,\ \{\ [\ let^\circ\, y = \{\ z_2\ \}\ in\ v\ ]\ \}\ \rangle\ ] \end{aligned}$$

by Eq. (26), (25) and (27),

$$\simeq_{ax} let\ \langle\ x, z_2\ \rangle = z_4\ in\ [\ \langle\ u\ ,\ \ let^\circ\ \ y = \{\ z_2\ \}\ in\ v\ \rangle\ ] : M(B \times D)$$

by Eq. (67). The map $t_{A,C}; M(f \times g)$ is equal to

$$z_4 : A \times MC \vdash \left(\begin{array}{l} \mathit{let}\ z_5 = (\mathit{let}\ \langle\ x, z_2\ \rangle = z_4\ \mathit{in}\ [\ \langle\ x, \{\ z_2\ \}\ \rangle\ ])\ \mathit{in} \\ [\ \mathit{let}^\circ\ z_3 = \{\ z_5\ \}\ \mathit{in}\ \mathit{let}\ \langle\ x, y\ \rangle = z_3\ \mathit{in}\ \langle\ u, v\ \rangle\ ] \end{array} \right)$$

$$\simeq_{ax} \mathit{let}\ \langle\ x, z_2\ \rangle = z_4\ \mathit{in} \left[\begin{array}{l} \mathit{let}^\circ\ z_3 = \langle\ x, \{\ z_2\ \}\ \rangle\ \mathit{in} \\ \mathit{let}\ \langle\ x, y\ \rangle = z_3\ \mathit{in}\ \langle\ u, v\ \rangle \end{array} \right] \quad \text{by Eq. (67),}$$

$$\simeq_{ax} \mathit{let}\ \langle\ x, z_2\ \rangle = z_4\ \mathit{in}\ [\ \mathit{let}^\circ\ y = \{\ z_2\ \}\ \mathit{in}\ \langle\ u, v\ \rangle\ ]$$

by Eq. (36) and (26),

$$\simeq_{ax} \mathit{let}\ \langle\ x, z_2\ \rangle = z_4\ \mathit{in}\ [\ \langle\ u, \mathit{let}^\circ\ y = \{\ z_2\ \}\ \mathit{in}\ v\ \rangle\ ] : M(B \times D)$$

by Eq. (70), which is equal to the previous map. Therefore t is a natural transformation.

The monadic equations of Definition 2.13 hold.

Equation (44) is treated as follows. The map $M\mu_A$ is equal to

$$y : M^3A \vdash [\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ [\ \{\ \{\ x\ \}\ \}\ ]\ ] : M^2A.$$

Then $M\mu_A; \mu_A$ is equal to

$$\begin{array}{ll} y : M^3A \vdash \mathit{let}\ z = [\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ [\ \{\ \{\ x\ \}\ \}\ ]\ ]\ \mathit{in}\ [\ \{\ \{\ z\ \}\ \}\ ] & \\ \quad \simeq_{ax} [\ \{\ \{\ [\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ [\ \{\ \{\ x\ \}\ \}\ ]\ ]\ \}\ \}\ ] & \text{by definition,} \\ \quad \simeq_{ax} [\ \{\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ [\ \{\ \{\ x\ \}\ \}\ ]\ \}\ ] & \text{by (67),} \\ \quad \simeq_{ax} [\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ \{\ [\ \{\ \{\ x\ \}\ \}\ ]\ \}\ ] & \text{by (68),} \\ \quad \simeq_{ax} [\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ \{\ \{\ x\ \}\ \}\ ] & \text{by (67),} \\ \quad \simeq_{ax} [\ \{\ \{\ \mathit{let}^\circ\ x = \{\ y\ \}\ \mathit{in}\ x\ \}\ \}\ ] & \text{by (68) twice,} \\ \quad \simeq_{ax} [\ \{\ \{\ \{\ y\ \}\ \}\ \}\ ] : MA & \text{by (33),} \end{array}$$

and the map $\mu_{MA}; \mu_A$ is equal to

$$\begin{array}{l} y : M^3A \vdash \mathit{let}\ z = [\ \{\ \{\ y\ \}\ \}\ ]\ \mathit{in}\ [\ \{\ \{\ z\ \}\ \}\ ] \\ \qquad = [\ \{\ \{\ [\ \{\ \{\ y\ \}\ \}\ ]\ \}\ \}\ ] \\ \qquad \simeq_{ax} [\ \{\ \{\ \{\ y\ \}\ \}\ \}\ ] : MA \quad \text{by (67).} \end{array}$$

They are equal. Equation (45) is treated in a similar manner.

To show that the monad M is commutative and strong, we have to examine the equations of Definition 2.14. We prove the commutativity of the monad. The upper path of Equation (49) is the judgment

$$x : MA \times MB \vdash \left(\begin{array}{l} \mathit{let}\ x_1 = (\mathit{let}\ \langle\ y, z\ \rangle = x\ \mathit{in}\ [\ \langle\ y, \{\ z\ \}\ \rangle\ ])\ \mathit{in} \\ [\ \mathit{let}^\circ\ x_2 = \{\ x_1\ \}\ \mathit{in}\ \mathit{let}\ \langle\ y, z\ \rangle = x_2\ \mathit{in}\ [\ \langle\ \{\ y\ \}, z\ \rangle\ ]\ ] \end{array} \right)$$

$$\simeq_{ax} \left[\begin{array}{l} \mathit{let}^\circ\ x_2 = \langle\ \pi_1 x, \{\ \pi_2 x\ \}\ \rangle\ \mathit{in} \\ \mathit{let}\ \langle\ y, z\ \rangle = x_2\ \mathit{in}\ [\ \langle\ \{\ y\ \}, z\ \rangle\ ] \end{array} \right] \quad \text{by (67),}$$

$$\simeq_{ax} \left[\begin{array}{l} \mathit{let}^\circ\ y = \pi_1 x\ \mathit{in} \\ \mathit{let}^\circ\ z = \{\ \pi_2 x\ \}\ \mathit{in}\ [\ \langle\ \{\ y\ \}, z\ \rangle\ ] \end{array} \right] \quad \text{by (69),}$$

$$\simeq_{ax} [\, [\, \langle\, \{\, \pi_1 x \,\}, \{\, \pi_2 x \,\} \,\rangle \,]\,] : M^2(A \times B) \quad \text{by (26) and (71).}$$

The lower path of Equation (49) is symmetric:

$$x : MA \times MB \vdash \left(\begin{array}{l} let\ x_1 = (let\ \langle\, y, z \,\rangle = x\ in\ [\, \langle\, \{\, y \,\}, z \,\rangle \,])\ in \\ [\, let^\circ\ x_2 = \{\, x_1 \,\}\ in\ let\ \langle\, y, z \,\rangle = x_2\ in\ [\, \langle\, y, \{\, z \,\} \,\rangle \,]\,] \end{array} \right)$$

$$\simeq_{ax} \left[\begin{array}{l} let^\circ\ x_2 = \langle\, \{\, \pi_1 x \,\}, \pi_2 x \,\rangle\ in \\ let\ \langle\, y, z \,\rangle = x_2\ in\ [\, \langle\, y, \{\, z \,\} \,\rangle \,] \end{array} \right] \quad \text{by (67),}$$

$$\simeq_{ax} \left[\begin{array}{l} let^\circ\ y = \{\, \pi_1 x \,\}\ in \\ let^\circ\ z = \pi_2 x\ in\ [\, \langle\, y, \{\, z \,\} \,\rangle \,] \end{array} \right] \quad \text{by (69),}$$

$$\simeq_{ax} [\, [\, \langle\, \{\, \pi_1 x \,\}, \{\, \pi_2 x \,\} \,\rangle \,]\,] : M^2(A \times B) \quad \text{by (26) and (70).}$$

So both paths are equal. Equations (46), (48) and (47) are treated similarly.

$\mathcal{C}_l(A, MB)$ is a module over $\mathcal{A}$. As an example of how this is shown, we give the case $\alpha \cdot (f + g) = \alpha \cdot f + \alpha \cdot g$, for maps $f, g : A \to MB$ in $\mathcal{C}_l$. If f is $x : A \vdash u : MB$ and g is $x : A \vdash v : MB$, then $\alpha \cdot (f + g)$ is the judgment

$$x : A \vdash [\, \alpha \cdot \{\, [\, \{\, u \,\} + \{\, v \,\} \,]\, \} \,] : MB.$$

The map $\alpha \cdot f + \alpha \cdot g$ is the judgment

$$x : A \vdash [\, \{\, [\, \alpha \cdot \{\, u \,\} \,]\, \} + \{\, [\, \alpha \cdot \{\, v \,\} \,]\, \} \,] : MB.$$

These two judgments are equivalent, using Eq. (67), (72), (73) and the algebraic rules of Table 3.

Finally, the fact that the bifunctor $\times$ and the composition of maps in $\mathcal{C}_{lM}$ are module homomorphisms is a direct consequence of the rules in Table 4.

This closes the proof of Theorem 2.33. □

Theorem 2.34. The category $\mathcal{C}_l$ is an $\mathcal{A}$-enriched computational category.

— The bifunctor $\Rightarrow$ is the type operator $\Rightarrow$, and it is defined on maps as follows:

$$(x : A \vdash u : B) \Rightarrow (y : C \vdash v : D) \quad = \quad z : B \Rightarrow C \vdash \Lambda x.\ let^\circ\ y = zu\ in\ v : A \Rightarrow D.$$

The bijection $\Psi_\Rightarrow$ is defined as follows:

$$\begin{array}{rcl} \Psi_\Rightarrow(x : A \times B \vdash u : MC) & = & y : A \vdash \Lambda z.\ let\ x = \langle\, y, z \,\rangle\ in\ \{\, u \,\} : B \Rightarrow C, \\ \Psi_\Rightarrow^{-1}(x : A \vdash u : B \Rightarrow C) & = & y : A \times B \vdash [\, let\ \langle\, x, z \,\rangle = y\ in\ uz \,] : MC. \end{array}$$

— The bifunctor $\to$ is the type operator $\to$, and it is defined on maps as follows:

$$\begin{array}{l} (x : A \vdash u : MB) \to (y : C \vdash v : MD) = \\ \qquad\qquad z : B \to C \vdash [\, \lambda x.\ let\ y = z\{\, u \,\}\ in\ \{\, v \,\} \,] : M(A \to D). \end{array}$$

The bijection $\Psi_\to$ is defined as follows:

$$\begin{array}{rcl} \Psi_\to(x : A \times B \vdash u : MC) & = & y : A \vdash [\, \lambda z.\ let\ x = \langle\, y, z \,\rangle\ in\ \{\, u \,\} \,] : M(B \to C), \\ \Psi_\to^{-1}(x : A \vdash u : M(B \to C)) & = & y : A \times B \vdash [\, let\ \langle\, x, z \,\rangle = y\ in\ \{\, u \,\} z \,] : MC. \end{array}$$

Proof. Note: given the naturality of $\Psi_{\Rightarrow}$ and $\Psi_{\rightarrow}$, the functoriality of $\Rightarrow$ and $\rightarrow$ comes for free. We first consider the map $\Psi_{\Rightarrow}$.

Bijectivity. Consider the map $f = x : A \times B \vdash u : MB$. Then $\Psi_{\Rightarrow}^{-1}\Psi_{\Rightarrow}(f)$ is equal to

$$\begin{aligned} x : A \times B \vdash & \;[\; \mathit{let}\ \langle\, y, z \,\rangle = x\ \mathit{in}\ (\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\ u\ \})z\ ] \\ & \simeq_{ax} [\; \mathit{let}\ \langle\, y, z \,\rangle = x\ \mathit{in}\ \mathit{let}\ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\ u\ \}\ ] \\ & \simeq_{ax} [\ \{\ u\ \}\ ] \\ & \simeq_{ax} u : MB, \end{aligned}$$

which is precisely f. Now, consider the map $g = x : A \vdash u : B \Rightarrow C$. Then $\Psi_{\Rightarrow}\Psi_{\Rightarrow}^{-1}(g)$ is equal to

$$\begin{aligned} y : A \vdash & \;\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\ [\ \mathit{let}\ \langle\, y, z \,\rangle = x\ \mathit{in}\ uz\ ]\ \} \\ & \simeq_{ax} \Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \mathit{let}\ \langle\, y, z \,\rangle = x\ \mathit{in}\ uz \\ & \simeq_{ax} \Lambda z.uz \\ & \simeq_{ax} u : B \Rightarrow B \quad \text{by (32).} \end{aligned}$$

This is equal to g: the map $\Psi_{\Rightarrow}$ is a bijection.
Naturality. Consider the base term f $= x : A' \times B' \vdash u : MC'$ and the three morphisms $g_A = z_A : A \vdash w_A : A'$, $g_B = z_B : B \vdash w_B : B'$ and $g_C = z_C : C' \vdash w_C : C$. Then

$$\begin{aligned} g_A \times g_B &= z_{AB} : A \times B \vdash \mathit{let}\ \langle\, z_A, z_B \,\rangle = z_{AB}\ \mathit{in}\ \langle\, w_A, w_B \,\rangle : A' \times B', \\ g_B \Rightarrow g_C &= z_{BC} : B' \Rightarrow C' \vdash \Lambda z_B.\ \mathit{let}^\circ\ \ z_C = z_{BC} w_B\ \mathit{in}\ w_C : B \Rightarrow C, \\ M g_C &= z_{MC} : MC' \vdash [\ \mathit{let}^\circ\ z_C = \{\ z_{MC}\ \}\ \mathit{in}\ w_C\ ] : MC. \end{aligned}$$

Now, $\Psi_{\Rightarrow}((g_A \times g_B); f; Mg_C)$ is equal to

$$\begin{aligned} z_A : A \vdash \Lambda z_B.\ \mathit{let}\ \ z_{AB} = \langle\, z_A, z_B \,\rangle\ \mathit{in} & \left\{ \begin{array}{l} \mathit{let}\ \langle\, z_A, z_B \,\rangle = z_{AB}\ \mathit{in} \\ \mathit{let}\ x = \langle\, w_A, w_B \,\rangle\ \mathit{in} \\ \mathit{let}\ z_{MC} = u\ \mathit{in} \\ [\ \mathit{let}^\circ\ z_C = \{\ z_{MC}\ \}\ \mathit{in}\ w_C\ ] \end{array} \right\} \\ \simeq_{ax} \Lambda z_B. & \left\{ \begin{array}{l} \mathit{let}\ \langle\, z_A, z_B \,\rangle = \langle\, z_A, z_B \,\rangle\ \mathit{in} \\ \mathit{let}\ x = \langle\, w_A, w_B \,\rangle\ \mathit{in} \\ \mathit{let}\ z_{MC} = u\ \mathit{in}\ [\ \mathit{let}^\circ\ z_C = \{\ z_{MC}\ \}\ \mathit{in}\ w_C\ ] \end{array} \right\} \\ \simeq_{ax} \Lambda z_B. & \left\{ \begin{array}{l} \mathit{let}\ x = \langle\, w_A, w_B \,\rangle\ \mathit{in} \\ \mathit{let}\ z_{MC} = u\ \mathit{in}\ [\ \mathit{let}^\circ\ z_C = \{\ z_{MC}\ \}\ \mathit{in}\ w_C\ ] \end{array} \right\} \\ \simeq_{ax} \Lambda z_B. & \left\{\ \mathit{let}\ x = \langle\, w_A, w_B \,\rangle\ \mathit{in}\ [\ \mathit{let}^\circ\ z_C = \{\ u\ \}\ \mathit{in}\ w_C\ ]\ \right\} \\ \simeq_{ax} \Lambda z_B. & \{\ [\ \mathit{let}\ x = \langle\, w_A, w_B \,\rangle\ \mathit{in}\ \mathit{let}^\circ\ z_C = \{\ u\ \}\ \mathit{in}\ w_C\ ]\ \} \\ \simeq_{ax} \Lambda z_B. & \ \mathit{let}\ \ x = \langle\, w_A, w_B \,\rangle\ \mathit{in}\ \mathit{let}^\circ\ z_C = \{\ u\ \}\ \mathit{in}\ w_C \\ \simeq_{ax} \Lambda z_B. & \ \mathit{let}^\circ\ \ z_C = (\mathit{let}\ x = \langle\, w_A, w_B \,\rangle\ \mathit{in}\ \{\ u\ \})\ \mathit{in}\ w_C : B \Rightarrow C. \end{aligned}$$

On the other hand, $g_A; (\Psi_{\Rightarrow} f); (g_B \Rightarrow g_C)$ is equal to

$$\begin{aligned}
z_A : A \vdash & \left(\begin{array}{l} let\ z_A = w_A\ in \\ let\ z_{BC} = (\Lambda y.\ let\ \ x = \langle\ z_A, y\ \rangle\ in\ \{\ u\ \})\ in \\ \Lambda z_B.\ let^\circ\ \ z_C = z_{BC} w_B\ in\ w_C \end{array} \right) \\
& \simeq_{ax} \left(\begin{array}{l} let\ z_{BC} = (\Lambda y.\ let\ \ x = \langle\ w_A, y\ \rangle\ in\ \{\ u\ \})\ in \\ \Lambda z_B.\ let^\circ\ \ z_C = z_{BC} w_B\ in\ w_C \end{array} \right) \\
& \simeq_{ax} \Lambda z_B.\ let^\circ\ \ z_C = (\Lambda y.\ let\ \ x = \langle\ w_A, y\ \rangle\ in\ \{\ u\ \}) w_B\ in\ w_C \\
& \simeq_{ax} \Lambda z_B.\ let^\circ\ \ z_C = (let\ x = \langle\ w_A, w_B\ \rangle\ in\ \{\ u\ \})\ in\ w_C : B \Rightarrow C.
\end{aligned}$$

Therefore, $\Psi_{\Rightarrow}((g_A \times g_B); f; Mg_C) = g_A; (\Psi_{\Rightarrow} f); (g_B \Rightarrow g_C)$: the bijection $\Psi_{\Rightarrow}$ is natural.

We now consider the map $\Psi_{\rightarrow}$.

Bijectivity. Consider the map $f = x : A \times B \vdash u : MB$. Then $\Psi_{\rightarrow}^{-1}\Psi_{\rightarrow}(f)$ is equal to

$$\begin{aligned}
x : A \times B \vdash & [\ let\ \langle\ y, z\ \rangle = x\ in\ \{\ [\ \lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ \{\ u\ \}\ ]\ \} z\ ] \\
& \simeq_{ax} [\ let\ \langle\ y, z\ \rangle = x\ in\ (\lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ \{\ u\ \}) z\ ] \\
& \simeq_{ax} [\ let\ \langle\ y, z\ \rangle = x\ in\ let\ x = \langle\ y, z\ \rangle\ in\ \{\ u\ \}\ ] \\
& \simeq_{ax} [\ \{\ u\ \}\ ] \\
& \simeq_{ax} u : MB,
\end{aligned}$$

which is precisely f. Now, consider the map $g = x : A \vdash u : M(B \to C)$. Then $\Psi_{\rightarrow}\Psi_{\rightarrow}^{-1}(g)$ is equal to

$$\begin{aligned}
y : A \vdash & [\ \lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ \{\ [\ let\ \langle\ y, z\ \rangle = x\ in\ \{\ u\ \} z\ ]\ \}\ ] \\
& \simeq_{ax} [\ \lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ let\ \langle\ y, z\ \rangle = x\ in\ \{\ u\ \} z\ ] \\
& \simeq_{ax} [\ \lambda z. \{\ u\ \} z\ ] \\
& \simeq_{ax} [\ \{\ u\ \}\ ] \\
& \simeq_{ax} u : M(B \to C).
\end{aligned}$$

This is equal to g: the map $\Psi_{\rightarrow}$ is a bijection.

Linearity. We show that $\Psi_{\rightarrow}$ is an $\mathcal{A}$-module homomorphism. Let $f = x : A \times B \vdash u : MC$ and $g = x : A \times B \vdash v : MC$ be two morphisms of $\mathcal{C}_l$. Then $\Psi_{\rightarrow}(f + \alpha \cdot g)$ is the by definition the map

$$y : A \vdash [\ \lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ \{\ [\ \{\ u\ \} + \alpha \cdot \{\ v\ \}\ ]\ \}\ ] : M(B \to C).$$

Applying Equation (67) and the linearity of the term constructs, this is axiomatically equivalent to

$$y : A \vdash \left[\begin{array}{l} (\lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ \{\ u\ \}) \\ +\alpha \cdot (\lambda z.\ let\ \ x = \langle\ y, z\ \rangle\ in\ \{\ v\ \}) \end{array} \right] : M(B \to C),$$

that is, $\Psi_{\rightarrow}(f) + \alpha \cdot \Psi_{\rightarrow}(g)$. Finally, since $0 \cdot f = \mathbf{0}$, the map $\Psi_{\rightarrow}$ preserves the zero of the module.

Naturality. Consider the base term $\mathrm{f} = x : A' \times B' \vdash u : MC'$ and the three morphisms $g_A = z_A : A \vdash w_A : MA'$, $g_B = z_B : B \vdash w_B : MB'$ and $g_C = z_C : C' \vdash w_C : MC$. Then the maps $g_A \times_M g_B$ and $g_B \to g_C$ are respectively

$$\begin{aligned}&z_{AB} : A \times B \vdash \mathit{let}\ \langle\, z_A, z_B \,\rangle = z_{AB}\ \mathit{in}\ [\, \langle\, \{\, w_A \,\}, \{\, w_B \,\} \,\rangle \,] : M(A' \times B'),\\ &z_{BC} : B' \to C' \vdash [\, \lambda z_B.\ \mathit{let}^\circ\ z_C = z_{BC}\{\, w_B \,\}\ \mathit{in}\ \{\, w_C \,\} \,] : M(B \to C).\end{aligned}$$

Now, $\Psi_\to((g_A \times_M g_B);_M f;_M g_C)$ is equal to

$$\begin{aligned}z_A : A \vdash &\left[\lambda z_B.\ \mathit{let}\ \ z_{AB} = \langle\, z_A, z_B \,\rangle\ \mathit{in} \left\{\begin{bmatrix} \mathit{let}\ \langle\, z_A, z_B \,\rangle = z_{AB}\ \mathit{in} \\ \mathit{let}^\circ\, x = \langle\, \{\, w_A \,\}, \{\, w_B \,\} \,\rangle\ \mathit{in} \\ \mathit{let}^\circ\, z_C = \{\, u \,\}\ \mathit{in}\ \{\, w_C \,\} \end{bmatrix}\right\}\right]\\ \simeq_{ax} &\left[\lambda z_B. \begin{pmatrix} \mathit{let}^\circ\, x = \langle\, \{\, w_A \,\}, \{\, w_B \,\} \,\rangle\ \mathit{in} \\ \mathit{let}^\circ\, z_C = \{\, u \,\}\ \mathit{in}\ \{\, w_C \,\} \end{pmatrix}\right] : M(B \to C)\end{aligned}$$

On the other hand, $g_A;_M (\Psi_\to f);_M (g_B \to g_C)$ is equal to

$$\begin{aligned}z_A : A \vdash &\begin{bmatrix} \mathit{let}^\circ\, z_A = \{\, w_A \,\}\ \mathit{in} \\ \mathit{let}^\circ\, z_{BC} = \{\, [\, \lambda y.\ \mathit{let}\ \ x = \langle\, z_A, y \,\rangle\ \mathit{in}\ \{\, u \,\} \,] \,\}\ \mathit{in} \\ \{\, [\, \lambda z_B.\ \mathit{let}^\circ\ z_C = z_{BC}\{\, w_B \,\}\ \mathit{in}\ \{\, w_C \,\} \,] \,\} \end{bmatrix}\\ \simeq_{ax} &\begin{bmatrix} \mathit{let}^\circ\, z_A = \{\, w_A \,\}\ \mathit{in} \\ \mathit{let}^\circ\, z_{BC} = (\lambda y.\ \mathit{let}^\circ\ \ x = \langle\, z_A, y \,\rangle\ \mathit{in}\ \{\, u \,\})\ \mathit{in} \\ \lambda z_B.\ \mathit{let}^\circ\ \ z_C = z_{BC}\{\, w_B \,\}\ \mathit{in}\ \{\, w_C \,\} \end{bmatrix}\\ \simeq_{ax} &\left[\lambda z_B. \begin{pmatrix} \mathit{let}^\circ\, z_A = \{\, w_A \,\}\ \mathit{in} \\ \mathit{let}^\circ\, z_{BC} = (\lambda y.\ \mathit{let}^\circ\ \ x = \langle\, z_A, y \,\rangle\ \mathit{in}\ \{\, u \,\})\ \mathit{in} \\ \mathit{let}^\circ\, z_C = z_{BC}\{\, w_B \,\}\ \mathit{in}\ \{\, w_C \,\} \end{pmatrix}\right]\end{aligned}$$

from Eq. (34),

$$\simeq_{ax} \left[\lambda z_B. \begin{pmatrix} \mathit{let}^\circ\, z_A = \{\, w_A \,\}\ \mathit{in} \\ \mathit{let}^\circ\, z_C = \begin{pmatrix} \mathit{let}^\circ\, y = \{\, w_B \,\}\ \mathit{in} \\ \mathit{let}^\circ\, x = \langle\, z_A, y \,\rangle\ \mathit{in}\ \{\, u \,\} \end{pmatrix}\ \mathit{in}\ \{\, w_C \,\} \end{pmatrix}\right]$$

from Eqs. (37) and (35),

$$\begin{aligned}\simeq_{ax} &\left[\lambda z_B. \begin{pmatrix} \mathit{let}^\circ\, z_A = \{\, w_A \,\}\ \mathit{in} \\ \mathit{let}^\circ\, y = \{\, w_B \,\}\ \mathit{in} \\ \mathit{let}^\circ\, x = \langle\, z_A, y \,\rangle\ \mathit{in}\ \ \mathit{let}^\circ\ \ z_C = \{\, u \,\}\ \mathit{in} \\ \{\, w_C \,\} \end{pmatrix}\right] \quad \text{from Eq. (37),}\\ \simeq_{ax} &\left[\lambda z_B. \begin{pmatrix} \mathit{let}^\circ\, x = \langle\, \{\, w_A \,\}, \{\, w_B \,\} \,\rangle\ \mathit{in} \\ \mathit{let}^\circ\, z_C = \{\, u \,\}\ \mathit{in}\ \{\, w_C \,\} \end{pmatrix}\right] : M(B \to C) \quad \text{from Eq. (36).}\end{aligned}$$

This proves that $\Psi_\to$ is a natural map. □

Notation 2.35. We use the following conventions.

— If $\Delta = (x_1 : A_1, \ldots, x_n : A_n)$ is a typing context, we write $\vec{\Delta}$ or $\vec{x}$ in place of the list of typed variables. Note that the list can be empty.

— The product of terms is extended to any finite product $\langle\, \vec{x}\,\rangle$. It is defined by induction:

$$\langle\ \rangle := *, \qquad \langle\, x\,\rangle := x, \qquad \langle\, y, \vec{x}\,\rangle := \langle\, y, \langle\, \vec{x}\,\rangle\,\rangle.$$

— The term construct π_i stands for the i-th projection of type $A_1 \times \cdots \times A_n$ $(i \leq n)$. Provided that $n \geq 2$, it is defined by induction:

$$\pi_1(s) := let\ \langle\, x, y\,\rangle = s\ in\ x, \qquad \pi_{n+2}s := let\ \langle\, x, y\,\rangle = s\ in\ \pi_{n+1}(y).$$

If $n = 1$, we define the only projection to be $\pi_1(u) = u$ and if $n = 0$, $\pi_1(u) = *$. In particular, we have $\pi_i\langle\, x_1, \ldots, x_n\,\rangle \simeq_{ax} x_i$, provided that $i \leq n$.

— We extend the notation $let\ \langle\, x, y\,\rangle = s\ in\ t$ of Notation 2.29 to tuples: Provided that $n \geq 1$,

$$let\ \langle\, x_1, \ldots, x_n\,\rangle = s\ in\ t \quad := \quad t[x_1 \leftarrow (\pi_1 x), \ldots, x_n \leftarrow (\pi_n x)].$$

Lemma 2.36. If we interpret the computational algebraic lambda-calculus in $\mathcal{C}_l$ then the equations

$$[\![\, x_1 : A_1, \ldots, x_n : A_n \vdash v : B\,]\!]^v \simeq_{ax} (y : A_1 \times \cdots \times A_n \vdash let\ \langle\, x_1, \ldots, x_n\,\rangle = y\ in\ v : B)$$

and

$$[\![\, x_1 : A_1, \ldots, x_n : A_n \vdash s : B\,]\!]^c$$
$$\simeq_{ax} (y : A_1 \times \cdots \times A_n \vdash [\ let\ \langle\, x_1, \ldots, x_n\,\rangle = y\ in\ s\ ] : MB)$$

hold.

Proof. Proof by induction on the derivation of the denotation. In the following we assume that $\Delta = (z_1 : D_1, \ldots, z_n : D_n)$.

Case $v = x$ and $v = *$. By definition of $let\ \langle - \rangle = -\ in\ -$.

Rule (52). By induction hypothesis, $f = [\![\, \Delta, x : A \vdash s : B\,]\!]^c$ is axiomatically equivalent to

$$y : \vec{D} \times A \vdash [\ let\ \langle\, \vec{z}, x\,\rangle = y\ in\ s\ ] : MB.$$

The map $\Psi_{\Rightarrow}(f)$ is by definition equal to

$$\begin{aligned} z' : \vec{D} \vdash \Lambda x.\ &let\ \ y = \langle\, z', x\,\rangle\ in\ \{\ [\ let\ \langle\, \vec{z}, x\,\rangle = y\ in\ s\ ]\ \} \\ &\simeq_{ax} \Lambda x.\ let\ \ y = \langle\, z', x\,\rangle\ in\ let\ \langle\, \vec{z}, x\,\rangle = y\ in\ s \\ &\simeq_{ax} \Lambda x.\ let\ \ \langle\, \vec{z}, x\,\rangle = \langle\, z', x\,\rangle\ in\ s \\ &= \Lambda x.\ let\ \ \langle\, \vec{z}\,\rangle = z'\ in\ let\ x = x\ in\ s \\ &= let\ \langle\, \vec{z}\,\rangle = z'\ in\ \Lambda x.\ let\ \ x = x\ in\ s \\ &= let\ \langle\, \vec{z}\,\rangle = z'\ in\ \Lambda x.s : A \Rightarrow B. \end{aligned}$$

Rule (53). By induction hypothesis, $f = [\![\, \Delta \vdash u : A_1 \times A_2\,]\!]^v$ is axiomatically equivalent to

$$y : \vec{D} \vdash let\ \langle\, \vec{z}\,\rangle = y\ in\ u : A_1 \times A_2.$$

The map $f; \pi_i$ is

$$y : \vec{D} \vdash let\ x = (let\ \langle\, \vec{z}\,\rangle = y\ in\ u)\ in\ \pi_i(x)$$

$$= let\ \langle\ \vec{z}\ \rangle = y\ in\ \pi_i(u) : A_i.$$

Rule (54). By induction hypothesis, $f = [\![\,\Delta \vdash u : A\,]\!]^v$ and $g = [\![\,\Delta \vdash v : B\,]\!]^v$ are respectively axiomatically equivalent to

$$y : \vec{D} \vdash let\ \langle\ \vec{z}\ \rangle = y\ in\ u : A,$$
$$y : \vec{D} \vdash let\ \langle\ \vec{z}\ \rangle = y\ in\ v : B.$$

The map $\langle\ f, g\ \rangle$ is

$$\begin{aligned} y : \vec{D} \vdash &\langle\ let\ \langle\ \vec{z}\ \rangle = y\ in\ u,\ \ let\ \ \langle\ \vec{z}\ \rangle = y\ in\ v\ \rangle \\ &= let\ \langle\ \vec{z}\ \rangle = y\ in\ \langle\ u, v\ \rangle : A \times B. \end{aligned}$$

Rule (55). By induction hypothesis, $f = [\![\,\Delta \vdash u : A\,]\!]^v$ is axiomatically equivalent to

$$y : \vec{D} \vdash let\ \langle\ \vec{z}\ \rangle = y\ in\ u : A.$$

The map $f; \eta_A$ is

$$\begin{aligned} y : \vec{D} \vdash &let\ x = (let\ \langle\ \vec{z}\ \rangle = y\ in\ u)\ in\ [\ x\ ] \\ &= [\ let\ \langle\ \vec{z}\ \rangle = y\ in\ u\ ] : MA. \end{aligned}$$

Rule (56). By induction hypothesis, $f = [\![\,\Delta, x : A \vdash s : B\,]\!]^c$ is axiomatically equivalent to

$$y : \vec{D} \times A \vdash [\ let\ \langle\ \vec{z}, x\ \rangle = y\ in\ s\ ] : MB.$$

The map $\Psi_{\rightarrow}(f)$ is by definition equal to

$$\begin{aligned} z' : \vec{D} \vdash &[\ \lambda x.\ let\ \ y = \langle\ z', x\ \rangle\ in\ \{\ [\ let\ \langle\ \vec{z}, x\ \rangle = y\ in\ s\ ]\ \}\ ] \\ &\simeq_{ax} [\ \lambda x.\ let\ \ y = \langle\ z', x\ \rangle\ in\ let\ \langle\ \vec{z}, x\ \rangle = y\ in\ s\ ] \\ &\simeq_{ax} [\ \lambda x.\ let\ \ \langle\ \vec{z}, x\ \rangle = \langle\ z', x\ \rangle\ in\ s\ ] \\ &= [\ \lambda x.\ let\ \ \langle\ \vec{z}\ \rangle = z'\ in\ let\ x = x\ in\ s\ ] \\ &= [\ let\ \langle\ \vec{z}\ \rangle = z'\ in\ \lambda x.\ let\ \ x = x\ in\ s\ ] \\ &= [\ let\ \langle\ \vec{z}\ \rangle = z'\ in\ \lambda x.s\ ] : M(A \rightarrow B). \end{aligned}$$

Rule (57). By induction hypothesis, $f = [\![\,\Delta \vdash s : A \Rightarrow B\,]\!]^c$ and $g = [\![\,\Delta \vdash t : A\,]\!]^c$ are respectively axiomatically equivalent to

$$y : \vec{D} \vdash [\ let\ \langle\ \vec{z}\ \rangle = y\ in\ s\ ] : M(A \Rightarrow B),$$
$$y : \vec{D} \vdash [\ let\ \langle\ \vec{z}\ \rangle = y\ in\ t\ ] : MA.$$

The map $\varepsilon_{A,B} = \Psi_{\Rightarrow}^{-1}(id_{A \Rightarrow B})$ corresponds to

$$y : (A \Rightarrow B) \times A \vdash [\ let\ \langle\ x, z\ \rangle = y\ in\ xz\ ] : MB.$$

The map $\langle\ f, g\ \rangle; s_{A \Rightarrow B, A}; M(\varepsilon_{A,B}); \mu_B$ is

$$y : \vec{D} \vdash \left(\begin{array}{l} let\ x_1 = \langle\ [\ let\ \langle\ \vec{z}\ \rangle = y\ in\ s\ ], [\ let\ \langle\ \vec{z}\ \rangle = y\ in\ t\ ]\ \rangle\ in \\ let\ x_2 = (let\ \langle\ y_1, y_2\ \rangle = x_1\ in\ [\ \langle\ \{\ y_1\ \}, \{\ y_2\ \}\ \rangle\ ])\ in \\ let\ x_3 = [\ let^{\circ}\ y = \{\ x_2\ \}\ in\ [\ let\ \langle\ x, z\ \rangle = y\ in\ xz\ ]\ ]\ in\ [\ \{\ \{\ x_3\ \}\ \}\ ] \end{array} \right)$$

$$\begin{aligned}
&\simeq_{ax} \left(\begin{array}{l} \mathit{let}\ x_2 = [\ \langle\ (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s), (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t)\ \rangle\ ]\ \mathit{in} \\ \mathit{let}\ x_3 = [\ \mathit{let}^\circ\ y = \{\ x_2\ \}\ \mathit{in}\ [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ]\ ]\ \mathit{in}\ [\ \{\ \{\ x_3\ \}\ \}\ ] \end{array}\right) \\
&\simeq_{ax} \left(\begin{array}{l} \mathit{let}\ x_3 = \left[\begin{array}{l} \mathit{let}^\circ\ y = \langle\ (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s), (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t)\ \rangle\ \mathit{in} \\ [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ] \end{array}\right]\ \mathit{in} \\ [\ \{\ \{\ x_3\ \}\ \}\ ] \end{array}\right) \\
&= \left[\left\{\left\{\left[\begin{array}{l} \mathit{let}^\circ\ y = \langle\ (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s), (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t)\ \rangle\ \mathit{in} \\ [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ] \end{array}\right]\right\}\right\}\right] \\
&\simeq_{ax} \left[\left\{\begin{array}{l} \mathit{let}^\circ\ y = \langle\ (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s), (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t)\ \rangle\ \mathit{in} \\ [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ] \end{array}\right\}\right] \\
&\simeq_{ax} \left[\begin{array}{l} \mathit{let}^\circ\ y = \langle\ (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s), (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t)\ \rangle\ \mathit{in} \\ \{\ [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ]\ \} \end{array}\right] \\
&\simeq_{ax} \left[\begin{array}{l} \mathit{let}^\circ\ y = \langle\ (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s), (\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t)\ \rangle\ \mathit{in} \\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz \end{array}\right] \\
&= [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ \mathit{let}^\circ\ y = \langle\ s,t\ \rangle\ \mathit{in}\ (\pi_1 y)(\pi_2 y)\ ] \\
&\simeq_{ax} [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ st\ ] : MB \qquad \text{by Eq. (69).}
\end{aligned}$$

Rule (58). By induction hypothesis, $f = [\![\,\Delta \vdash s : A \to B\,]\!]^c$ and $g = [\![\,\Delta \vdash t : A\,]\!]^c$ are respectively axiomatically equivalent to

$$\begin{aligned}
y : \vec{D} &\vdash [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ] : M(A \to B), \\
y : \vec{D} &\vdash [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t\ ] : MA.
\end{aligned}$$

The map $\overline{\varepsilon}_{A,B} = \Psi_{\to}^{-1}(\eta_{A\to B})$ corresponds to

$$y : (A \to B) \times A \vdash [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ \{\ [\ x\ ]\ \}z\ ] : MB,$$

that is,

$$y : (A \to B) \times A \vdash [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ] : MB.$$

The map $\langle\ f,g\ \rangle; s_{A\to B,A}; M(\overline{\varepsilon}_{A,B}); \mu_B$ is

$$y : \vec{D} \vdash \left(\begin{array}{l} \mathit{let}\ x_1 = \langle\ [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ], [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t\ ]\ \rangle\ \mathit{in} \\ \mathit{let}\ x_2 = (\mathit{let}\ \langle\ y_1, y_2\ \rangle = x_1\ \mathit{in}\ [\ \langle\ \{\ y_1\ \}, \{\ y_2\ \}\ \rangle\ ])\ \mathit{in} \\ \mathit{let}\ x_3 = [\ \mathit{let}^\circ\ y = \{\ x_2\ \}\ \mathit{in}\ [\ \mathit{let}\ \langle\ x,z\ \rangle = y\ \mathit{in}\ xz\ ]\ ]\ \mathit{in} \\ [\ \{\ \{\ x_3\ \}\ \}\ ] \end{array}\right) : MB,$$

which is precisely the same term as in the previous case. It is therefore axiomatically equivalent to $y : \vec{D} \vdash [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ st\ ] : MB$, using the same sequence of arguments.

Rule (59). By induction hypothesis, $f = [\![\,\Delta \vdash s : A_1 \times A_2\,]\!]^c$ is axiomatically equivalent to

$$y : \vec{D} \vdash [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ] : M(A_1 \times A_2).$$

The map $f; (M\pi_i)$ is

$$\begin{aligned}
y : \vec{D} \vdash\ & \mathit{let}\ x_1 = [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ]\ \mathit{in}\ [\ \mathit{let}^\circ\ x = \{\ x_1\ \}\ \mathit{in}\ \pi_i(x)\ ] \\
&\simeq_{ax} \mathit{let}\ x_1 = [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ]\ \mathit{in}\ [\ \pi_i(\{\ x_1\ \})\ ] \quad \text{by Eq. (39) or (40),}
\end{aligned}$$

$$
\begin{aligned}
&= [\ \pi_i(\{\ [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ]\ \})\ ] \\
&\simeq_{ax} [\ \pi_i(\mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s)\ ] \\
&= [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ \pi_i(s)\ ] : MA_i.
\end{aligned}
$$

Rule (60). By induction hypothesis, $f = [\![\ \Delta \vdash s : A\ ]\!]^c$ and $g = [\![\ \Delta \vdash t : B\ ]\!]^c$ are respectively axiomatically equivalent to

$$
\begin{aligned}
&y : \vec{D} \vdash [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ] : MA, \\
&y : \vec{D} \vdash [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t\ ] : MB.
\end{aligned}
$$

The map $\langle\ f, g\ \rangle; s_{A,B}$ is

$$
\begin{aligned}
y : \vec{D} \vdash &\left(\begin{array}{l} \mathit{let}\ x_1 = \langle\ [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s\ ], [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t\ ]\ \rangle\ \mathit{in} \\ \mathit{let}\ \langle\ z_1, z_2\ \rangle = x\ \mathit{in}\ [\ \langle\ \{\ z_1\ \}, \{\ z_2\ \}\ \rangle\ ] \end{array}\right) \\
&\simeq_{ax} [\ \langle\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ s,\ \ \mathit{let}\ \ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ t\ \rangle\ ] \\
&= [\ \mathit{let}\ \langle\ \vec{z}\ \rangle = y\ \mathit{in}\ \langle\ s, t\ \rangle\ ] : M(A \times B).
\end{aligned}
$$

This closes the list of cases and the proof of Lemma 2.36. □

Theorem 2.37 (Completeness). If we interpret the computational algebraic lambda-calculus in $\mathcal{C}_l$ then the equations

$$[\![\ x : A \vdash v : B\ ]\!]^v \simeq_{ax} (x : A \vdash v : B) \quad \text{and} \quad [\![\ x : A \vdash t : B\ ]\!]^c \simeq_{ax} (x : A \vdash [\ t\ ] : MB)$$

hold.

Proof. The theorem is an easy corollary of Lemma 2.36. Indeed, the typing context consists of a single variable: $[\![\ x : A \vdash v : B\ ]\!]^v \simeq_{ax} (x : A \vdash \mathit{let}\ \langle\ x\ \rangle = x\ \mathit{in}\ v : B)$. By Rule (26), this is also v. Now, $[\![\ x : A \vdash t : B\ ]\!]^c \simeq_{ax} (x : A \vdash \mathit{let}\ \langle\ x\ \rangle = x\ \mathit{in}\ [\ t\ ] : MB)$. Again by Rule (26), this is axiomatically equivalent to $[\ t\ ]$. □

Corollary 2.38. Two base terms are axiomatically equivalent if and only if for any $\mathcal{A}$-enriched computational category their denotations are equal.

Proof. If two typing judgments are axiomatically equivalent, by Theorem 2.28 they have equal denotations in any $\mathcal{A}$-enriched computational category. Now, suppose they do have equal denotations in any $\mathcal{A}$-enriched computational category. In particular, this is verified in $\mathcal{C}_l$. Therefore, they are axiomatically equivalent by Theorem 2.37. □

2.5. *Consistency*

We are now in position to state the consistency of the equational description of Section 2.1.

Theorem 2.39. The typing judgments $\Delta \vdash \mathbf{0} : A$ and $\Delta \vdash u : A$ (where u is a base term) are not axiomatically equivalent.

Proof. In **Set**, the denotation of the former is the zero map $\mathbf{0} : [\![\ \Delta\ ]\!] \to M[\![\ A\ ]\!]$, whereas the denotation of the latter a non-zero function. □

3. Adding controlled divergence

Because of Theorem 2.39, the term Y_b of Equation (1) is not constructable in the computational algebraic lambda-calculus. In this section, we add to the semantics and to the language a notion of fixpoint in order to understand what goes wrong in the untyped system.

3.1. *A fixpoint operator*

We first describe what is a fixpoint in the categorical model. A fixpoint is a computation, so we define it in the Kleisli category.

Definition 3.1. With the notations of Definition 2.19, a *fixpoint combinator* in an $\mathcal{A}$-enriched computational category $\mathcal{C}$ is a mapping $\mathbf{Y}_{X,Y} : \mathcal{C}(X \times MY, MY) \to \mathcal{C}(X, MY)$ such that for every morphism $f : X \times MY \to MY$, the diagram

$$\begin{tikzcd} X \arrow[r, "\mathbf{Y}_{X,Y} f"] \arrow[d, "\langle\, id_X, id_X \,\rangle"'] & MY \\ X \times X \arrow[r, "X \times \mathbf{Y}_{X,Y} f"'] & X \times MY \arrow[u, "f"'] \end{tikzcd} \tag{75}$$

is commutative. When the context is clear, we write $\mathbf{Y}$ in place of $\mathbf{Y}_{X,Y}$. We also transparently use the curried version of the operator $\mathbf{Y}$ sending $\mathcal{C}(X, MY \Rightarrow Y)$ to $\mathcal{C}(X, MY)$.

Example 3.2. The set-based model of Section 2.3.1 can be equipped with a fixpoint combinator, by taking $\mathbf{Y}$ to be the trivial mapping. Remember that in this setting, $\mathcal{C}(X, MY)$ is a singleton.

In Example 3.2, the monad is the trivial one. This turns out to be a general fact and we can show that if an $\mathcal{A}$-enriched computational category has a fixpoint combinator, then the monad is trivial.

Theorem 3.3. An $\mathcal{A}$-enriched computational category $\mathcal{C}$ has a fixpoint combinator if and only if all the maps in $\mathcal{C}_M$ are identified.

Proof. It is enough to show that between any two objects A and B of $\mathcal{C}_M$ there is exactly one map. First, there is always the map $\mathbf{0} : A \to_{\mathcal{C}} MB$. Then, consider any other map $f : A \to MB$ and build the map $g : A \times MB \to MB$ as the sum of $\pi_2 : A \times MB \to MB$ and $g_2 = \pi_1; f : A \times MB \to MB$. The commutative diagram satisfied by $\mathbf{Y}g$ can be rewritten as the equality $\mathbf{Y}g = \mathbf{Y}g + f$, that is, $0 \cdot \mathbf{Y}g = f$. Since $0 \cdot \mathbf{Y}g = \mathbf{0}$, the map f is equal to $\mathbf{0}$. □

3.2. *Recasting the equational theory*

In the proof of Theorem 3.3, the part that makes all the maps in $\mathcal{C}_M$ collapse is the fact that $0 \cdot f = \mathbf{0}$, for all f. In order to retain consistency, a natural solution is therefore to remove the property $0 \cdot f = \mathbf{0}$. Instead of a module, we only require the homset $\mathcal{C}_M(X, Y)$ to be a *weak* module.

Definition 3.4. Given a ring $\mathcal{A}$, a *weak $\mathcal{A}$-module* $(M,+,0,\cdot)$ is the data consisting of a commutative monoid $(M,+,0)$ and an operation $(\cdot):\mathcal{A}\times M\to M$ such that for all a,b in $\mathcal{A}$ and for all x,y in M,

$$a\cdot(x+y)=a\cdot x+a\cdot y, \qquad r\cdot(s\cdot x)=(rs)\cdot x,$$
$$(a+b)\cdot x=a\cdot x+b\cdot x, \qquad 1\cdot x=x.$$

Remark 3.5. Note that the equations are the same as the ones in Definition 3.4. This time however, since M is only a monoid, i.e. $v+((-1)\cdot v)=0\cdot v\neq 0$.

Definition 3.6. A *weak $\mathcal{A}$-enriched computational category* is the same structure as in Definition 2.19, apart from the fact that the Kleisli category $\mathcal{C}_M$ is not enriched anymore over $\mathcal{A}$-modules but over *weak* $\mathcal{A}$-modules.

A weak $\mathcal{A}$-enriched computational category has a *fixpoint combinator* if there is a mapping $\mathbf{Y}_{X,Y}$ satisfying the same properties as in Definition 3.1.

Lemma 3.7. Any $\mathcal{A}$-enriched computational category is a weak $\mathcal{A}$-enriched computational category.

Proof. Any $\mathcal{A}$-module is also a weak $\mathcal{A}$-module. $\square$

3.3. *A concrete model*

We modified the semantics, implicitly claiming that a weak $\mathcal{A}$-enriched category with fixpoints does not necessarily have a trivial monad. In this section, we provide a concrete example of such a category. We will discuss in Section 3.6 the notion of convergence it provides.

3.3.1. *A lattice-based framework.* Consider a lattice $(X,\leq)$. A subset S of X is called *directed* if for every two elements x,y in S there is an element z in S such that $x\leq z$ and $y\leq z$. The lattice $(X,\leq)$ is called *directed complete* if every directed subset admits a least upper bound.

A *partial monoid* is a family of directed-complete lattices $X=\{(X_i,\leq_i)\}_{i\in I}$. We identify X with $\uplus_{i\in I}X_i$, the disjoint union of all the X_i's, and we write $\leq_X$ (or $\leq$ when the context is clear) for the induced ordering on this union.

Let $X=\{(X_i,\leq_i)\}_{i\in I}$ and $Y=\{(Y_j,\leq_j)\}_{j\in J}$ be two partial monoids. A *partial-monoid homomorphism* $f:X\to Y$ is a function $\uplus X_i\to\uplus Y_j$, monotone on the induced ordering and preserving limits of directed sets. Note that, due to its monotonicity f induces a set-map $\tilde{f}:I\to J$ such that if $x\in X_i$, then $f(x)\in Y_{\tilde{f}(i)}$.

We call **PMon** the category of partial monoids and partial-monoid homomorphisms.

Theorem 3.8. The category **PMon** can be made into a weak $\mathcal{A}$-enriched computational category with fixpoints.

3.3.2. *Cartesian structure.* The category **PMon** is cartesian with the following structure.

— The terminal object is the family $\{(\{\star\},\leq)\}$ where $\{\star\}$ is a singleton and $\leq$ is the trivial relation.

— The product of $X = \{(X_i, \leq_i)\}_{i\in I}$ and $Y = \{(Y_j, \leq_j)\}_{j\in J}$ is the partial monoid $X \times Y = \{(X_i \times Y_j, \leq_{i,j})\}_{(i,j)\in I\times J}$ where $(x, y) \leq_{i,j} (x', y')$ if and only if $x \leq_i x'$ and $y \leq_j y'$.

— Let $f : X \to Y$ and $g : X \to Z$ be two morphisms of **PMon**. We define $\langle\, f, g\,\rangle$ as the set-function $x \mapsto \langle\, f(x), g(x)\,\rangle$. This map is trivially a morphism of **PMon**.

— The projections $\pi_1 : X \times Y \to X$ and $\pi_2 : X \times Y \to Y$ are the set-functions

$$\begin{array}{rccc}
\pi_1 : & \uplus_{(i,j)\in I\times J}(X_i \times Y_j) & \longrightarrow & \uplus_{i\in I}X_i \\
 & (x_i, y_j) & \longmapsto & x_i \\
\pi_2 : & \uplus_{(i,j)\in I\times J}(X_i \times Y_j) & \longrightarrow & \uplus_{j\in J}Y_j \\
 & (x_i, y_j) & \longmapsto & y_j
\end{array}$$

where we assume that $x_i \in X_i$ and $y_j \in Y_j$, for $i \in I$ and $j \in J$.

— Equations (41), (42) and (43) are satisfied since the structure is inherited from the cartesian structure of **Set**. Similarly, the uniqueness of $\langle\, f, g\,\rangle$ is inherited from the one of **Set**.

3.3.3. *A powerset construction.* We adapt Hoare powerdomains (Winskell, 1983), also called lower powerdomains (Heckmann, 1990, Ch.18) to partial monoids. Because of its operational meaning in the language, we shall write $\sum X$ for the powerset of X instead of the more standard $\mathcal{P}(X)$.

Let $\{(X_i, \leq_i)\}_{i\in I}$ be a partial monoid. If $x \in X_i$ we write $\downarrow x$ for the set of all the elements in X_i smaller or equal to x. If $S \subseteq X$, we write $\downarrow S$ for the directed closure of the union of all the sets $\downarrow x$, when x ranges over S. We define $\sum X$ to be the set

$$\{\downarrow S \;\mid\; S \subseteq X\}.$$

The ordering relation $\leq_{\sum X}$ on $\sum X$ is the subset-ordering relation. The join operation is the union, and the meet operation is the intersection. Note that $(\sum X, \leq)$ is directed complete.

3.3.4. *A commutative, strong monad.* We now define an operator M on partial monoids sending X to the partial monoid consisting of only one lattice $MX = \{(\sum X, \leq_{\sum X})\}$. We extend this operator to a functor on **PMon** by defining the image of a morphism $f : X \to Y$ to Mf, defined as follows:

$$Mf(S) = \downarrow \{f(x) \mid x \in S\} \tag{76}$$

Together with the three maps

$$\begin{array}{ccc}
\eta_X : X \to MX & \mu_X : MMX \to MX & t_{X,Y} : X \times MY \to M(X \times Y) \\
x \mapsto \downarrow x, & S \mapsto \downarrow \bigcup_{T\in S} T, & (x, S) \mapsto \downarrow \{(x, y) \mid y \in S\}
\end{array}$$

it defines a commutative, strong monad on **PMon**: These three maps are natural transformations in **PMon**, and it is possible to show that they satisfy the required equations.

3.3.5. *Homset.* The set of morphisms $\mathbf{PMon}(X, MY)$ can be endowed with a structure of lattice: $(f \vee g)(x) = f(x) \vee g(x)$ and $(f \wedge g)(x) = f(x) \wedge g(x)$. The lattice is directed complete since MY is directed complete: given an directed set S of morphisms in $\mathbf{PMon}(X, MY)$, the least upper bound of S is the morphism sending x to $\downarrow \cup_{f \in S} f(x)$. The set of morphisms $\mathbf{PMon}(X, MY)$ is therefore a partial monoid (as the union of only one directed complete lattice). It is an object of $\mathbf{PMon}$, and we can define a map

$$\begin{array}{rccc} \Psi_{\Rightarrow}: & \mathbf{PMon}(X \times Y, MZ) & \to & \mathbf{PMon}(X, \mathbf{PMon}(Y, MZ)) \\ & f & \mapsto & (x \mapsto (y \mapsto f(x,y))). \end{array}$$

We claim that this map is well defined. The map $\Psi_{\Rightarrow}$ is a bijection: Given any morphism g in $\mathbf{PMon}(X, \mathbf{PMon}(Y, MZ))$, the map $f = \Psi_{\Rightarrow}^{-1}(g) : X \times Y \to MZ$ is defined by $f(x,y) = g(x)(y)$, and one can show that this map is a morphism of $\mathbf{PMon}$. Finally, the naturality of $\Psi_{\Rightarrow}$ is inherited from the one of $\mathbf{Set}$.

3.3.6. *Closure of the Kleisli category.* Let X and Y be any two partial monoids of the form $\{(X_i, \leq_i)\}_{i \in I}$ and $\{(Y_j, \leq_j)\}_{j \in J}$. Define the new partial monoid $X \to Y$ as the family of lattices

$$\{\mathbf{PMon}(X_i, Y_j)\}_{(i,j) \in I \times J}.$$

If $f : X \to MY$ is a morphism of $\mathbf{PMon}$, let $\sigma f \in M(X \to Y)$ be

$$\downarrow \bigcup_{i,j} \{\ g : X_i \to Y_j \mid \forall x \in X_i,\ g(x) \in f(x)\ \},$$

and if $S \in M(X \to Y)$, let $\tau S \in \mathbf{PMon}(X, MY)$ be the map sending some $x \in X_i$ to the element of MY defined by

$$\downarrow \{\ g(x) \mid g \in S \text{ and } g \text{ has domain } X_i\ \}.$$

We claim that

$$\begin{array}{rccc} \Psi_{\to}: & \mathbf{PMon}(X \times Y, MZ) & \to & \mathbf{PMon}(X, M(Y \to Z)) \\ & f & \mapsto & (x \mapsto \sigma(y \mapsto f(x,y))) \end{array}$$

is a natural bijection. Its inverse sends $g \in \mathbf{PMon}(X, M(Y \to Z))$ to the morphism mapping (x,y) to $\tau(y \mapsto g(x,y))$.

3.3.7. *A structure of weak $\mathcal{A}$-module.* We can define a structure of weak $\mathcal{A}$-module on the lattice $\mathbf{PMon}(X, MY)$ by setting $f + g = f \vee g$ and $\alpha \cdot f = f$. The zero element is the function sending all x's to $\perp$, the bottom element. We get a weak $\mathcal{A}$-module since one does not require $0 \cdot f$ to be equal to the zero element. The resulting structure is a weak $\mathcal{A}$-enriched computational category.

To keep up with the weak module interpretation, in the rest of the paper, an object MX in $\mathbf{PMon}$ will be considered as a set of (formal) sums rather than as a set of subsets.

3.3.8. *Fixpoint.* We complete the sketch of the proof of Theorem 3.8 by exhibiting a fixpoint combinator. Given the partial monoid $X = \{(X_i, \leq_i)\}_{i \in I}$, the partial monoid MX is simply a directed complete lattice. If $\perp$ is its bottom element, the usual technique works for showing that the operator Y defined as sending $f : MX \to MX$ to the least upper bound of the directed subset $\{f^n(\bot) \mid n \in \mathbb{N}\}$ is a fixpoint operator and a morphism $\mathbf{PMon}(MX, MX) \to MX$ of **PMon**. We refer the reader to *e.g.* (Plotkin, 1983) for the proof.

3.4. *Adding a fixpoint to the algebraic lambda-calculus*

3.4.1. *A fixpoint operator.* We now turn to the question of modifying the interpretation of Section 2.1 to account for a fixpoint. We add to the language a unary term operator Y satisfying the typing rule

$$\Delta \vdash s : MA \Rightarrow A \quad \text{implies} \quad \Delta \vdash Y(s) : A, \tag{77}$$

verifying the axiomatic relation

$$Y(v) \simeq_{ax} v[\, Y(v)\,] \tag{78}$$

where v is a base term (the rule is a translation of Equation (75)), and linear with respect to the module structure.

3.4.2. *Inconsistency.* It is possible to build a term Y_b with the behavior of Equation (1):

$$Y_b := Y\Lambda x.(b + \{\, x\,\}). \tag{79}$$

Indeed, $Y\Lambda x.(b + \{\, x\,\})$ is equivalent to the term $(\Lambda x.(b + \{\, x\,\}))[\, Y\Lambda x.(b + \{\, x\,\})\,]$, which is in turn equivalent to $b + Y\Lambda x.(b + \{\, x\,\})$. Provided that $\Delta \vdash b : B$, the typing judgment $\Delta \vdash Y_b : B$ is valid. Of course, if we keep the equational theory of Section 2.1, the system becomes as inconsistent as with the untyped calculus.

3.4.3. *The zero in the algebra of terms* In the light of the analysis of Section 3.1, we have the required tools to understand what goes wrong. Consider the typing judgment $x : MA \vdash x - x : MA$. With the equational system of Section 2.1, this typing judgment is equivalent to $x : MA \vdash \mathbf{0} : MA$. We claim that this interpretation is correct as long as the term x "does not contain any potential infinity". With the additional construct Y, we can replace x with $[\, Y_a\,]$ (where Y_a is constructed as in Equation (79)) for some term a of type A. Consider the two terms

$$(\lambda y.*)((\lambda x.(x - x))[\, Y_a\,]) \;\; (80) \qquad (\lambda y.\{\, y\,\})((\lambda x.(x - x))[\, Y_a\,]) \;\; (81).$$

Term (80) is equivalent to $(\lambda y.*)(0 \cdot [\, Y_a\,])$ and then to $0 \cdot *$. It is reasonable to think that this is equivalent to $\mathbf{0}$, thus making $0 \cdot [\, Y_a\,]$ also equivalent to $\mathbf{0}$. Term (81), on the contrary, is equivalent to $Y_a - Y_a$, the flawed term of Equation (2).

As discussed in Section 3.2, the problem does not show up when writing the equation $[\, Y_a\,] - [\, Y_a\,] = 0 \cdot [\, Y_a\,]$ but when one equates it with $\mathbf{0}$. The term $0 \cdot [\, Y_a\,]$ is a "weak zero". It makes a computation "null" as long as it does not diverge (but there is always a diverging term of any inhabited type by using the construction (79)). Therefore, despite the fact that $\mathcal{A}$ is a ring, the set of terms of the form $\alpha \cdot s$ for a fixed term s is only a commutative monoid: addition does not admit an inverse, it only has an identity element $0 \cdot s$. This is consistent with previous studies (Vaux, 2009; Selinger, 2003).

3.5. *Recasting the language*

We can recast the computational algebraic lambda-calculus to match the solution provided in Section 3.2 as follows.

Definition 3.9. Consider the typed language of Definition 2.1 augmented with a fixpoint combinator Y. The definition of base terms in unchanged: $Y(s)$ is always considered as a computation. The typing rules of Table 2 are augmented with Rule (77).

The axiomatic equivalence is still coming from the Tables 3, 4 and 5, with the following modifications:

— Rule (3) (stating $0 \cdot u \simeq_{ax} \mathbf{0}$) is removed;
— Rule (78) (stating $Y(v) \simeq_{ax} v[\, Y(v)\,]$ when v is a base term) is added.
— A rule relating λ-abstractions and the term construct Y is added:

$$(\lambda x.Y(x))s \ \simeq_{ax}\ Y(s). \tag{82}$$

Let us call this language the *weak algebraic computational lambda-calculus* and the corresponding category of base terms $\mathcal{C}_l^w$.

A term in the weak algebraic lambda-calculus can be encoded in any weak $\mathcal{A}$-enriched computational category $\mathcal{C}$ with fixpoints using the same rules as for the regular algebraic lambda-calculus, augmented with the rule

$$\frac{[\![\, \Delta \vdash s : MA \Rightarrow A \,]\!]^c \ = \ [\![\, \Delta \,]\!] \xrightarrow{f} M(M[\![\, A \,]\!] \Rightarrow [\![\, A \,]\!])}{[\![\, \Delta \vdash Y(s) : A \,]\!]^c \ = \ [\![\, \Delta \,]\!] \xrightarrow{f;_M \mathbf{Y}(id_{MA\Rightarrow A})} M[\![\, A \,]\!]} \tag{83}$$

It is possible to transpose the soundness and completeness results of Section 2.4 to this new situation.

Theorem 3.10. The following results are correct.

1 The denotation of the weak computational algebraic lambda-calculus in weak $\mathcal{A}$-enriched computational categories with fixpoints is sound.
2 $\mathcal{C}_l^w$ is a weak $\mathcal{A}$-enriched computational category with fixpoints. The required structure is the same as the one in $\mathcal{C}_l$, plus the $\mathbf{Y}$ operator, defined by

$$\mathbf{Y}(x : A \times MB \vdash u : MB) \quad = \quad y : A \vdash [\, Y \Lambda z.\ let\ \ x = \langle\, y, z \,\rangle\ in\ \{\ u\ \}\,] : MB.$$

3 The weak computational algebraic lambda-calculus is an internal language for weak $\mathcal{A}$-enriched computational categories with fixpoints.

Proof. We can almost take the exact same proofs of the corresponding theorems of Section 2; the only differences are:

— in Rule (3), and this rule is only used in the proof of the fact that $\mathcal{C}_M$ is enriched over $\mathcal{A}$-module. If we remove the rule, we can only show that is enriched over the category of $\mathcal{A}$-weak-modules, which is the only thing required.
— with respect to the addition of the fixpoint combinator Y.

1 In the proof of soundness we have to check that $[\![\, \Delta \vdash v[\, Y(v)\,] : A \,]\!]^c$ is equal to

the map $[\![\, \Delta \vdash Y(v) : A \,]\!]^c$, for a base term $\Delta \vdash v : MA \Rightarrow A$. Provided that the b-denotation of v is f, its c-denotation is $f;\eta$ and therefore

$$[\![\, \Delta \vdash v[\, Y(v)\,] : A \,]\!]^c = \langle\, f;\eta\ ,\ f;\mathbf{Y}(id);\eta \,\rangle_M;_M \varepsilon.$$

By naturality of $\Psi_{\Rightarrow}$, this is equal to $f;_M \mathbf{Y}(id)$, that is, $[\![\, \Delta \vdash Y(v) : A \,]\!]^c$.
We also have to check that Rule (82) is valid. But this is clear by naturality of $\Psi_{\to}$.

2 We have to check that $\mathcal{C}_l$ indeed has fixpoints. Provided that f is the map $x : A \times MB \vdash v : MB$, the morphism $\langle\, id, id \,\rangle; (X \times (\mathbf{Y}f)); f$ is equal to

$$\begin{aligned} y : A \vdash\ & \mathit{let}\ x = \langle\, y, [\, Y\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\, v \,\}\,]\,\rangle\ \mathit{in}\ v \\ & \simeq_{ax} \mathit{let}\ x = \langle\, y, [\, Y\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\, v \,\}\,]\,\rangle\ \mathit{in}\ [\,\{\, v \,\}\,] \\ & = [\, \mathit{let}\ x = \langle\, y, [\, Y\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\, v \,\}\,]\,\rangle\ \mathit{in}\ \{\, v \,\}\,] \\ & \simeq_{ax} [\, (\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\, v \,\})[\, Y\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\, v \,\}\,]\,] \\ & \simeq_{ax} [\, Y\Lambda z.\ \mathit{let}\ \ x = \langle\, y, z \,\rangle\ \mathit{in}\ \{\, v \,\}\,] \quad \text{by Eq. (78),} \end{aligned}$$

which is the morphism $\mathbf{Y}f$.

3 To prove that the weak computational algebraic lambda-calculus is an internal language for weak $\mathcal{A}$-enriched computational categories, it is enough to show that in $\mathcal{C}_l^w$, the denotation $[\![\, \vec{z} : \vec{D} \vdash Y(s) : A \,]\!]^c$ is equal to the map $x : \vec{D} \vdash [\, \mathit{let}\ \langle\, \vec{z}\,\rangle = x\ \mathit{in}\ Y(s)\,] : MA$.
First note that $\mathbf{Y}(id_{MA\Rightarrow A}) : (MA \Rightarrow A) \to MA$, the map sending a function to its fixpoint is in $\mathcal{C}_l^w$ the morphism

$$x : MA \Rightarrow A \vdash [\, Y(x)\,] : MA.$$

Now, suppose that $\vec{z} : \vec{D} \vdash s : MA \Rightarrow A$. By induction hypothesis its c-denotation in $\mathcal{C}_l^w$ is the map $x : \vec{D} \vdash [\, \mathit{let}\ \langle\, \vec{z}\,\rangle = x\ \mathit{in}\ s\,] : M(MA \Rightarrow A)$. The denotation of $Y(s)$ is the map $f;_M \mathbf{Y}(id)$, which is equal to

$$\begin{aligned} x : \vec{D} \vdash\ & [\, \mathit{let}\ y = [\, \mathit{let}\ \langle\, \vec{z}\,\rangle = x\ \mathit{in}\ s\,]\ \mathit{in}\ [\, Y(y)\,]\,] \\ & \simeq_{ax} [\, \mathit{let}\ \langle\, \vec{z}\,\rangle = x\ \mathit{in}\ (\lambda y.Y(y))s\,] \\ & \simeq_{ax} [\, \mathit{let}\ \langle\, \vec{z}\,\rangle = x\ \mathit{in}\ Y(s)\,] : MA \quad \text{by Eq. (82).} \end{aligned}$$

Therefore, the interpretation of any term of the weak computational lambda-calculus in $\mathcal{C}_l^w$ is axiomatically equivalent to itself.

This concludes the proof of Theorem 3.10. □

3.6. *Examples*

Consider the category **PMon** defined in Section 3.3. Being a weak $\mathcal{A}$-enriched computational category, it defines a model in which one can interpret the computational algebraic lambda-calculus. In the following examples, we shall use the types *bit* and *int*. Term constants *tt* and *ff*, accounting for the boolean values true and false, and a term constant $\bar{0}$, accounting for the zero of the natural numbers, are added to the language. We also add a unary term operator *succ* to account for the successor function.

The denotation of *bit* is the trivial partial monoid $\{\mathit{tt}\} \cup \{\mathit{ff}\}$, the denotation of *int* is the enumerable union of the trivial partial monoids $\{\overline{n}\}$ when n spans the natural numbers. The images of the corresponding term constructs are the ones coming from their interpretation in **Set**.

Example 3.11. The denotation of $M(\mathit{int})$ is the lattice based on the set of (possibly infinite) sums $\sum_{i\in I} \bar{i}$ (using the symbol $\sum$ to describe the subset $\{\bar{i} \mid i \in I\}$). Its bottom element is 0 and its top element the sum of all integers.

Example 3.12. Consider the term

$$Y\Lambda x.(\bar{0} + \mathit{succ}\{\ x\ \}). \tag{84}$$

of type *int*. With n uses of Rule (78) we can show that it is equivalent to

$$\left(\sum_{i=0}^{n-1} \bar{i}\right) + \mathit{succ}^n(Y\Lambda x.(\bar{0} + \mathit{succ}\{\ x\ \})).$$

We describe the denotation of the term (84) in **PMon**. The denotation of $\Lambda x.(\bar{0} + \mathit{succ}\{\ x\ \})$ being the morphism $f : M\mathbb{N} \to M\mathbb{N}$ defined by

$$f(\sum_{i\in U} \bar{i}) = \bar{0} + \sum_{i\in U} \bar{i},$$

the element $f^n(\mathbf{0})$ is equal to

$$\sum_{i=0}^{n-1} \bar{i}.$$

The least upper bound of the sequence $\{f^n(\mathbf{0}) \mid n \in \mathbb{N}\}$ is therefore $\sum_{i=0}^{\infty} \bar{i}$, that is, the sum of all numbers.

Example 3.13. In **PMon**, the denotation of the judgment $x : M(\mathit{int}) \vdash \{\ x\ \} : \mathit{int}$ is the identity function. The term $Y\Lambda x.\{\ x\ \} : \mathit{int}$ has therefore the denotation $\mathbf{0} \in M(\mathbb{N})$. It has the same denotation as the term $x : M(\mathit{int}) \vdash \mathbf{0} : \mathit{int}$.

The judgment $x : M(\mathit{int}) \vdash \{\ x\ \} + \bar{0} : \mathit{int}$ has for denotation the map sending a linear combination of integers x to $x + \bar{0}$. Therefore, $Y_{\bar{0}} = Y\Lambda x.(\{\ x\ \} + \bar{0}) : \mathit{int}$ has for denotation $\bar{0}$.

Remark 3.14. From Examples 3.12 and 3.13, we see that **PMon** does not distinguish between a "truly" looping term and the term **0**. It can however distinguish various non-normalizing terms.

3.7. *Consistency of the equational theory*

Since we have at least one concrete instance of weak $\mathcal{A}$-enriched computational category with fixpoints, we are able to state the consistency of the equational description.

Theorem 3.15. The typing derivations $\Delta \vdash \mathbf{0} : bit$, $\Delta \vdash \mathit{ff} : bit$, $\Delta \vdash 0 \cdot Y_{\mathit{tt}} : bit$ and $\Delta \vdash \mathit{ff} + Y_{\mathit{tt}} : bit$ are not axiomatically equivalent.

Proof. In **PMon**, the denotations of the four judgments are constant functions: the first maps to $\perp$, the second maps to ff, the third to tt and the last one to $\mathit{tt} + \mathit{ff}$. $\square$

Remark 3.16. One can however note that $\Delta \vdash 0 \cdot Y_{\mathit{tt}} : bit$ and $\Delta \vdash \mathit{tt} + 0 \cdot Y_{\mathit{tt}} : bit$ have the same denotation tt in **PMon**. This is consistent with the fact that they are axiomatically equivalent.

4. Discussion

4.1. *Call-by-name and call-by-value fragments*

We started the discussion in Section 1.2 by analyzing two vectorial lambda-calculi, one dubbed call-by-value and the other one call-by-name. In this section, we give simply typed version of these two languages and relate them to the computational algebraic lambda-calculus and its categorical semantics.

4.1.1. *Call-by-value fragment.* The language is built on the sets of terms and values

$$\begin{array}{rcl} s,t & ::= & x \mid \lambda x.s \mid st \mid s+t \mid \alpha \cdot s \mid \mathbf{0}, \\ u,v & ::= & x \mid \lambda x.s. \end{array}$$

A simple type system is

$$A,B \quad ::= \quad \iota \mid A \to B.$$

The equivalence $\simeq_{ax}^{\text{cbv}}$ on terms consists of the algebraic and call-by-value rules of Table 1.

Let $(-)^{\#}$ be the mapping of terms of the vectorial call-by-value lambda-calculus into the computational algebraic lambda-calculus, defined as follows.

$$\begin{array}{ccc} x^{\#} = x, & (\lambda x.s)^{\#} = \Lambda x.s^{\#}, & (st)^{\#} = s^{\#}t^{\#}, \\ (s+t)^{\#} = s^{\#} + t^{\#}, & (\alpha \cdot s)^{\#} = \alpha \cdot s^{\#}, & \mathbf{0}^{\#} = \mathbf{0}. \end{array}$$

Types are mapped as follows.

$$\iota^{\#} = \iota, \qquad (A \to B)^{\#} = (A^{\#} \Rightarrow B^{\#}).$$

Lemma 4.1. If $\Delta \vdash s : A$ is a valid vectorial call-by-value typing judgment, then $\Delta^{\#} \vdash s^{\#} : A^{\#}$ is a valid typing judgment in the computational algebraic lambda-calculus.

Proof. Proof by induction on the derivation of $\Delta \vdash s : A$. $\square$

Lemma 4.2. If $\Delta \vdash s \simeq_{ax}^{\text{cbv}} t : A$, then $\Delta^{\#} \vdash s^{\#} \simeq_{ax} t^{\#} : A^{\#}$. $\square$

Using Lemma 4.2, we have a sound denotation of a typing judgment of the vectorial call-by-value calculus in any $\mathcal{A}$-enriched computational category, by interpreting its image under $(-)^{\#}$. In particular, the denotation of a value $x : A \vdash v : B$ in the vectorial call-by-value lambda-calculus is a morphism $[\![A]\!] \to [\![B]\!]$ in $\mathcal{C}$, and the denotation of a term

$x : A \vdash s : B$ a map $[\![A^{\#}]\!] \to [\![B^{\#}]\!]$ in the Kleisli category. The language lives in the sub-structure consisting of

— the cartesian category $(\mathcal{C}, \times, \top)$,
— the strong, commutative monad M whose Kleisli category is Kleisli-closed and enriched over $\mathcal{A}$-modules.

This is the structure of regular Moggi's computational lambda-calculus, together with an enrichment over $\mathcal{A}$-modules.

4.1.2. *Call-by-name fragment.* The language is built on the set of terms

$$s, t \quad ::= \quad x \mid \lambda x.s \mid st \mid s + t \mid \alpha \cdot s \mid \mathbf{0},$$

A simple type system is

$$A, B \quad ::= \quad \iota \mid A \to B.$$

The equivalence $\simeq_{ax}^{\text{cbn}}$ on terms consists of the algebraic and call-by-name rules of Table 1.

Let $(-)^{\%}$ be the mapping of terms of the vectorial call-by-name lambda-calculus into the computational algebraic lambda-calculus, defined as follows.

$$x^{\%} = \{\, x \,\}, \qquad (\lambda x.s)^{\%} = \lambda x.s^{\%}, \qquad (st)^{\%} = s^{\%}[\, t^{\%} \,],$$
$$(s+t)^{\%} = s^{\%} + t^{\%}, \qquad (\alpha \cdot s)^{\%} = \alpha \cdot s^{\%}, \qquad \mathbf{0}^{\%} = \mathbf{0}.$$

Types are mapped as follows.

$$\iota^{\%} = \iota, \qquad (A \to B)^{\%} = (M(A^{\%}) \to B^{\%}).$$

As for the call-by-value case, we can formulate two lemmas relating the vectorial call-by-name lambda-calculus and the computational algebraic lambda-calculus.

Lemma 4.3. If $x_1 : B_1, \ldots, x_n : B_n \vdash s : A$ is a valid vectorial call-by-name typing judgment, then $x_1 : MB_1^{\%}, \ldots, x_n : MB_n^{\%} \vdash s^{\#} : MA^{\#}$ is a valid typing judgment in the computational algebraic lambda-calculus.

Proof. Proof by induction on the derivation of $\Delta \vdash s : A$. The proof is slightly less easy than Lemma 4.1 since the translation is a bit more involved. □

Lemma 4.4. If $x_1 : B_1, \ldots, x_n : B_n \vdash s \simeq_{ax}^{\text{cbn}} t : A$, then $x_1 : MB_1^{\%}, \ldots, x_n : MB_n^{\%} \vdash s^{\%} \simeq_{ax} t^{\%} : M(A^{\%})$. □

Using Lemma 4.4, as for the call-by-value case, we can derive a sound denotation of a typing judgment of the vectorial call-by-name lambda-calculus in any $\mathcal{A}$-enriched computational category by interpreting its image under $(-)^{\%}$. A term $x : A \vdash s : B$ of the vectorial call-by-name calculus is a morphism $M[\![A^{\%}]\!] \to M[\![B^{\%}]\!]$ in $\mathcal{C}$. In this case, the only structure required for interpreting the language is the Kleisli category $(\mathcal{C}_M, \times, \to, \top)$ with its enriched structure. Note that the bifunctor $\Rightarrow$ is not used.

4.2. *Other vectorial lambda-calculi*

A few existing works discuss the question of vectorial lambda-calculi. In this section we list them and compare them with our approach.

4.2.1. *The quantum lambda-calculus of van Tonder (2004).* In this line of work, the question is to encode an untyped lambda-calculus directly onto quantum bits and to find a unitary maps that acts as a call-by-value reduction on terms. In other words, the lambda-terms are thought of as base elements of some Hilbert space where the reduction would be some unitary map. In order to be able to do regular quantum computation, and to be able to create linear combinations of terms, the lambda-calculus is equipped with constants tt and ff to stand for booleans and other constants to stand for unitary maps. For example, if H is the constant standing for the Hadamard gate, the term $H\,\mathit{tt}$ should reduce to the linear combination $\frac{1}{\sqrt{2}}(\mathit{tt}+\mathit{ff})$.

Forcing the reduction to be unitary turns out to be too strong to get non-trivial linear combinations of terms: All terms in superposition need to be equal as strings of symbols up to tt and ff, therefore making the language fall back on a classical lambda-calculus with pointers.

This line of work does neither consider the question of the denotation nor discuss the eventuality of fixpoints. The former is supposed to be understood through the quantum interpretation and the latter treated by the unitarity of the reduction.

4.2.2. *The language Lineal of Arrighi and Dowek (2008).* This algebraic lambda-calculus also has a quantum flavor. The idea behind this work is to forget about the unitarity of the reduction: unlike the language in (van Tonder, 2004), in Lineal are considered general superpositions of terms. The goal is to get some insights on the computational power of a generalized vectorial call-by-value language that can have any terms in superposition. The language keeps a bit of quantumness in the way it deals with distributivity of the vectorial structure over terms constructs. The lambda-abstraction is not distributive: a lambda-abstraction is thought of as being the description of an operator, and should therefore be duplicated "as it". It behaves as the Λ-abstraction in the computational algebraic lambda-calculus. The application is distributive on the left and on the right, keeping the same spirit as (van Tonder, 2004).

The main question that is addressed by this work is to find a confluent rewrite system of the untyped calculus that forbid the behavior of Equation 1. Interestingly enough, it is possible to modify the rewrite system in order to keep all the rewrite rules concerned with the module structure.

In summary, this work is careful to forbid any diverging term to reduce, for keeping the equational theory consistent.

4.2.3. *The algebraic lambda-calculus of (Vaux, 2009).* Derived from the differential lambda-calculus developed in (Ehrhard and Regnier, 2003), this calculus is very close to the call-by-name lambda-calculi described in Sections 1.2 and 4.1. However, the fact that a rewrite system is considered raises problems when dealing with normalization of terms.

In this work, the problem described in Equation (2) is solved by taking the ring of scalar to be a semi-ring, but Rule (3) is not forbidden. Instead, the semiring is asked to be *positive*, that is, $a + b = 0$ if and only if $a = b = 0$. We do not enforce this condition for the computational algebraic lambda-calculus, even in the presence of fixpoints.

Finally, although fixpoints are allowed in the untyped version of the calculus, (Vaux, 2009) is not concerned with the question of the semantics of the calculus.

4.2.4. *Probabilistic and non-deterministic calculi.* Many lambda-calculi such as (Jones and Plotkin, 1989; Bucciarelli et al., 2009) are concerned with semantical issues and possess probabilistic or non-deterministic properties. These calculi can be seen as having vectorial constructs based on a module over the semi-ring of positive reals for probabilistic effects, and the semi-ring $\{0, 1\}$ for non-deterministic effects.

However, the semi-rings $\mathbb{R}^+$ and $\{0, 1\}$ share the very peculiar property of being positive, simplifying the problem arising in Equation (2) and making the languages closer to the language of (Vaux, 2009) than to a language with a full ring of scalars.

Also, in many cases (e.g. Jones and Plotkin, 1989), being based on the computational meta-language of (Moggi, 1991) the calculi do not consider the lambda-abstraction as being distributive over the vectorial structure, making the models slightly unsuitable for the computational algebraic calculus, even if we were restricting the ring of scalars to a suitable semi-ring.

4.3. *Characterization of scalars and finer convergence*

The categorical analysis we developed is fine enough to allow the model **PMon** to distinguish between between various infinitary behaviors.

However, this model does not consider the scalars at all. It would be nice to be able to distinguish between $2 \cdot *$ and $3 \cdot *$. It would also be nice to take into account a possible topology on the ring of scalars. For example, consider the term

$$Y \Lambda x.(* + \frac{1}{2}\{\ x\ \}) : \top.$$

If we were working in the field of reals or the field of complexes, it would be nice to be able to equate this term with $2 \cdot *$. The category **PMon** is far from being able to account for such a notion of limits on the ring of scalars.

5. Conclusion

In this paper, we describe an algebraic, simply-typed computational lambda-calculus and we derive a categorical semantics. We provide various concrete models, effectively showing the consistency of the equational description. We then focus on the addition of a fixpoint for this language, and we generalize the categorical description for this setting. We give a non-trivial concrete model with the interpretations of various terms. This shows that the categorical semantics is consistent and that there exist models in which one can distinguish between several divergent terms.

This raises the question of the complete description for the possible operational behaviors of the computational algebraic lambda-calculus.

6. Acknowledgments

I would like to thank Gilles Dowek for introducing me to algebraic calculi. I would also like to thank Pablo Arrighi and the research group CAPP in Grenoble for helpful discussions.

References

Pablo Arrighi and Alejandro Díaz-Caro. Scalar system F for linear-algebraic λ-calculus: Towards a quantum physical logic. In *Proceedings of QPL'09*, volume 270 number 2 of *Electronic Notes in Theoretical Computer Science*, pages 219–229, 2011.

Pablo Arrighi and Gilles Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 17–31, 2008.

Franco Barbanera and Maribel Fernández. Combining first and higher-order rewrite systems with type assignment systems. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA'93*, volume 664 of *Lecture Notes in Computer Science*, pages 60–74, 1993.

Frédéric Blanqui, Jean-Pierre Jouannaud, and Mitsuhiro Okada. The calculus of algebraic constructions. In *Proceedings of the 10th International Conference on Rewriting Techniques and Applications*, pages 301–316, London, UK, 1999. Springer-Verlag.

Val Breazu-Tannen and Jean Gallier. Polymorphic rewriting conserves algebraic strong normalization. *Theoretical Computer Science*, 83(1):3–28, 1991.

Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational model of a parallel and non-deterministic lambda-calculus. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, volume 5407 of *Lecture Notes in Computer Science*, pages 107–121. Springer Berlin / Heidelberg, 2009.

Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15 (4):615–646, 2005.

Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1–2):1–41, 2003.

Andrzej Filinski. Representing monads. In *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 446–457, 1996.

Reinhold Heckmann. *Power Domain Constructions.* PhD thesis, Universität der Saarlandes, Saabrücken, Germany, 1990.

Claire Jones and Gordon D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the 4th Symposium on Logic in Computer Science*, pages 186–195, 1989.

Gregory M. Kelly. *Basic Concepts of Enriched Category Theory*, volume 64 of *London Mathematical Society Lecture Notes Series.* Cambridge University Press, 1982. Available in Reprint in Theory and Application of Categories, No 10, 1982.

Joachim Lambek and Philip Scott. *Introduction to Higher Order Categorical Logic.* Cambridge University Press, 1989.

Saunders Mac Lane. *Categories for the Working Mathematician.* Springer Verlag, 1998.

Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

Benjamin C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

Gordon D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

Gordon D. Plotkin. Pisa notes (on domain theory). Unpublished set of notes, 1983.

Peter Selinger. Order-incompleteness and finite lambda-reduction models. *Theoretical Computer Science*, 309:43–63, 2003.

Benoît Valiron. Semantics of a typed algebraic lambda-calculus. In S. Barry Cooper, Prakash Panangaden, and Elham Kashefi, editors, *Proceedings of the 6th Workshop on Developments in Computational Models*, volume 26 of *Electronic Proceedings in Theoretical Computer Science*, pages 147–158, 2010.

André van Tonder. A lambda calculus for quantum computation. *SIAM Journal of Computing*, 33:1109–1135, 2004.

Lionel Vaux. The algebraic lambda-calculus. *Mathematical Structures in Computer Science*, 19:1029–1059, 2009.

Glynn Winskell. A note on powerdomains and modality In Marek Karpinski, editor, *Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 505–514, 1983.