A Kleisli-like construction for Parametric Monads, with Examples

University of Bath, Claverton Down Road, Bath. BA2 7AY,
wjg27@bath.ac.uk,
WWW home page: http://people.bath.ac.uk/wjg27

Abstract. Keywords: denotational semantics, category theory, game semantics

1 Introduction

1.1 Monads and Kleisli categories

This paper is about general frameworks for modelling effects in a categorical semantics. The seminal paper in this field is Moggi's *Computational Lambda-Calculus and Monads* [Moggi(1989)], which first put forward the idea that if $\mathcal{C}$ is a categorical model of a programming language, then we can simulate an effect in that language via a suitable *monad* on $\mathcal{C}$. Given such a monad, its Kleisli category $Kl_M\mathcal{C}$ is then a model of a language formed by adding the effect to the original language.

Spivey [Spivey(1990)] and Wadler [Wadler(1992),Wadler(1990)] have transferred this theory to the real world by using monads as programming tools, particularly in the Haskell programming language.

Recent progress in the field of game semantics, however, has shown that monads and Kleisli categories are not the be-all and end-all of modelling effects. There are perfectly game semantic models of effects such as state and nondeterminism that do not arise as the Kleisli category for a monad on some more basic category of games. These models are typically built up using intuition – for example, by relaxing the determinism condition on strategies to give a model of a nondeterministic language – and do not readily fit into any category theoretic framework the way Kleisli categories do. The goal of this paper is to examine some possible frameworks that will allow us to study these and other models in a systematic way.

A closely related problem is that of using a common technique to model a particular effect in different settings. Many of our favourite monads (e.g., the powerset monad on **Set** for nondeterminism) live in some particular category and cannot readily be set up in other categories.

Before we start, it is worth mentioning one example of a Kleisli category arising naturally in game semantics. If we read Ghica's work on *slot games* [Ghica(2005)], it is not difficult to show that the category he constructs is in fact isomorphic to the Kleisli category for the *reader monad* $\mathbb{C} \to \_$, where $\mathbb{C}$ is the game corresponding to a singleton set.

Since the reader monad is entirely built out of the Cartesian closed structure of the category, we can perform a similar trick in many different settings, although perhaps not as successfully as in game semantics.

1.2 Promonads

It is instructive at this point to see what happens if we generalize from functors to profunctors. We define a *promonad* on a category $\mathcal{C}$ to be a monoid in the monoidal category $\mathrm{Prof}[\mathcal{C},\mathcal{C}]$ of endo-profunctors on $\mathcal{C}$. This concept generalizes that of a monad, since an endofunctor $M : \mathcal{C} \to \mathcal{C}$ may be identified with the profunctor $\mathcal{C}[\_, M(\_)] : \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$.

Since the set $\mathcal{C}[A, MB]$ is precisely the set of Kleisli morphisms from A to B, it is natural to wonder whether we can perform a Kleisli-type construction on a promonad as well. it turns out that we can: if $\mathcal{D} : \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$ is a promonad, then we may define a category (also called $\mathcal{D}$) whose objects are the objects of $\mathcal{C}$, and where the morphisms from A to B are the elements of the set $\mathcal{D}(A,B)$. The promonad action

$$(\mathcal{D} \cdot \mathcal{D})(A,C) = \int^{B} \mathcal{D}(A,B) \times \mathcal{D}(B,C) \to \mathcal{D}(A,C)$$

gives us composition in our category and the unit

$$1(A,B) = \mathcal{C}[A,B] \to \mathcal{D}[A,B]$$

gives us a functor $\mathcal{C} \to \mathcal{D}$ (and, in particular, the identity morphisms).

In fact, the converse is true: if $\mathcal{C}$ and $\mathcal{D}$ are categories with the same objects and $J : \mathcal{C} \to \mathcal{D}$ is an identity-on-objects functor, then the functor $\mathcal{D}(JA, JB) = \mathcal{D}(A,B) : \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$ may be given the structure of a promonad on $\mathcal{C}$. Thus, the rather complicated definition of a promonad is in fact equivalent to the much simpler idea of an identity-on-objects functor. We shall therefore use the word 'promonad' to mean both things.

In one sense this is very exciting for us, because we can see that promonads generalize not only monads, but also a very large number of the models developed in Game Semantics. For example, the inclusion of the category of games and deterministic strategies into the category of games and nondeterministic strategies is the identity on objects, so it is a promonad. So is the inclusion of the category of games and innocent strategies into the category of games and visible strategies.

More generally, this fits in with the idea that effects should not normally create new types, but should instead give us more flexibility when it comes to writing programs – exactly what is modelled by an identity-on-objects functor.

On the other hand, the sheer generality of promonads means that they don't give us any ways to actually create new models, since the data required to specify one is the entire category that we are trying to build.

One way of describing the purpose of this work, then, is to discover classes intermediate between monads and promonads that are more general than pure monads but can still be specified in a concise way that allows us to transfer them between different settings.

1.3 Monads on presheaf categories

We can think of an endo-profunctor $\mathcal{F} : \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$ as a functor $\mathcal{C} \to [\mathcal{C}^{op}, \mathbf{Set}]$. Since the Yoneda embedding exhibits $[\mathcal{C}^{op}, \mathbf{Set}]$ as the free cocompletion of $\mathbf{Set}$, this can alternatively be characterized as a colimit-preserving endofunctor on $[\mathcal{C}^{op}, \mathbf{Set}]$. Similarly, any promonad on $\mathcal{C}$ can alternatively be considered as a monad on $[\mathcal{C}^{op}, \mathbf{Set}]$ whose underlying endofunctor preserves colimits.

1.4 Parametric monads

Melliès [Melliès(2012)], following work by Smirnov [Smirnov(2008)], has demonstrated that the concept of a lax monoidal category action is a useful generalization of that of a monad. A lax action of a monoidal category $\mathcal{C}'$ upon a category $\mathcal{C}$ is a lax monoidal functor $\mathcal{C}' \to \mathrm{End}[\mathcal{C}]$; monads are the special case when $\mathcal{C}'$ is trivial. For this reason, Melliès renames lax monoidal actions to 'parametric monads', where we think of the action of $\mathcal{C}'$ as a kind of monad on $\mathcal{C}$ that is parameterized by the objects of $\mathcal{C}'$.

This idea has been further generalized by Katsumata [Katsumata(2016)], who has given a definition of a 'Kleisli-like resolution' of a parametric monad. This particular construction, which takes a parametric monad on a category $\mathcal{C}$ parameterized by a monoidal category $\mathcal{C}'$ and yields a new category, generalizes many important properties of the Kleisli category – in particular, it gives us an adjunction with the original category – but it does not fit into our framework, since the objects of the category it creates are not the objects of the original category $\mathcal{C}$, but pairs of an object of $\mathcal{C}$ and an object of $\mathcal{C}'$.

Once again, it is useful to generalize to profunctors. First, if $\mathcal{C}$ is a category and $\mathcal{C}'$ a monoidal category, we define a *parametric promonad on $\mathcal{C}$ parameterized by $\mathcal{C}'$* to be a lax monoidal functor $\mathcal{C}' \to \mathrm{Prof}[\mathcal{C}, \mathcal{C}]$.

As with promonads, we can now look at what we have defined and try to recast it as a particular kind of category. Suppose that $\mathcal{D} : \mathcal{C}' \times \mathcal{C}^{op} \times \mathcal{C}$ is a parametric promonad on $\mathcal{C}$ parameterized by $\mathcal{C}'$. After some thought, we can see that $\mathcal{D}$ corresponds to a category whose objects are the objects of $\mathcal{C}$ and where the morphisms from A to B are labelled by objects of $\mathcal{C}'$ in such a way that the composition of a morphism labelled with X and a morphism labelled with Y is a morphism labelled with $X \otimes Y$.

One way to state this is as follows. Given a parametric promonad $\mathcal{D} : \mathcal{C}' \times \mathcal{C}^{op} \times \mathcal{C} \to \mathbf{Set}$, we define a bicategory $\mathcal{D}^2$:

- whose objects are the objects of $\mathcal{C}$,
- where the 1-morphisms from A to B are pairs (X, f), where X is an object of $\mathcal{C}'$ and $f \in \mathcal{D}(X, A, B)$

– and where the 2-morphisms from (X, f) to (Y, g) are morphisms $h: X \to Y$ in $\mathcal{C}'$ such that $\mathcal{D}(h, A, B)(f) = g$.

This bicategory comes equipped with two important bifunctors:

– a bifunctor $J: \mathcal{C} \to \mathcal{D}^2$ (where we consider $\mathcal{C}$ as a bicategory with only identity 2-morphisms;
– a bifunctor $l: \mathcal{D}^2 \to \mathcal{C}'$ given by the 'labelling' (considering $\mathcal{C}'$ as a bicategory with a single object).

In order to get a promonad resolution of the parametric promonad, then, we need to convert this bicategory into a 1-category $\mathcal{D}$, which we do by quotienting out the 1-morphisms in $\mathcal{D}^2$ by the action of the 2-morphisms: i.e., the smallest equivalence relation such that two 1-morphisms are related if there is a 2-morphism from one to the other.

More explicitly, the set of 1-morphisms from A to B in our new category will be given by the following colimit.

$$\mathcal{D}(A, B) = \varinjlim_{X : \mathcal{C}'} \mathcal{D}(X, A, B)$$

This category $\mathcal{D}$ now admits an identity-on-objects functor from $\mathcal{C}$, meaning that we have found a natural collapse from parametric promonads to ordinary promonads.

In the particular case that the parametric promonad is in fact a parametric *monad*, then we will call the category we obtain $\mathcal{C}/\mathcal{C}'$. The morphisms in $\mathcal{C}/\mathcal{C}'$ are given by

$$\mathcal{C}/\mathcal{C}'(A, B) = \varinjlim_{X : \mathcal{C}'} \mathcal{C}(A, X.B)\,,$$

with the composition of $f: A \to X.B$ and $g: B \to Y.C$ being given by the composite

$$A \xrightarrow{f} X.B \xrightarrow{X.g} X.Y.C \to (X \otimes Y).C\,.$$

Note that if $\mathcal{C}'$ is the trivial category (so that the action is in fact a monad), then the category $\mathcal{C}/\mathcal{C}'$ is the usual Kleisli category.

1.5 Effects, monads and colimits

The colimit that appears in the definition of the category $\mathcal{C}/\mathcal{C}'$ is instructive, because it is linked to the fact that many important monads may be expressed as colimits. For example, let **Surj** be the category of sets and surjections. Then, if A is a set, we have

$$\mathcal{P}(A) \cong \varinjlim_{X : \mathbf{Surj}^{op}} [X, A]\,,$$

naturally in A.

This is useful, because the body of the colimit – i.e., the function set $[X, A]$ – has a natural equivalent inside any Cartesian closed category that admits a functor out of **Set**. Moreover, the functor $[X, A]$ is actually an *action* of $\mathbf{Surj}^{op}$ (with the Cartesian product) upon the category of sets. So even though we cannot define the powerset *monad* inside arbitrary Cartesian closed categories, we can still define this action of $\mathbf{Surj}^{op}$.

Unfortunately, when we try to apply our construction to this action in the category of sets, we do not get the usual Kleisli category for the powerset monad. To see why, note that from the discussion above, we have

$$\begin{aligned}\mathbf{Set}/\mathbf{Surj}^{op}(A, B) &= \varinjlim_{X\,:\,\mathbf{Surj}^{op}} [A, [X, B]] \\ &\cong \varinjlim_{X\,:\,\mathbf{Surj}^{op}} [X, [A, B]] \\ &\cong \mathcal{P}([A, B])\,,\end{aligned}$$

while the morphisms in the Kleisli category take a different form:

$$\mathrm{Kl}_{\mathcal{P}}\,\mathbf{Set}(A, B) = [A, \mathcal{P}(B)]\,.$$

Nevertheless, there are still a few things we can take from this. The sets $\mathcal{P}([A, B])$ and $[A, \mathcal{P}(B)]$ are not equal, but there is still a natural function

$$\mathcal{P}([A, B]) \to [A, \mathcal{P}(B)]\,,$$

that sends a set $\mathcal{F}$ of functions to the function h (the *amalgamation* of $\mathcal{F}$) where

$$h(a) = \{f(a) :\ f \in \mathcal{F}\}\,.$$

This amalgamation function is not *quite* surjective: for example, suppose that $g :\ \{0, 1\} \to \mathcal{P}(\{0\})$ sends 0 to $\{0\}$ and 1 to the empty set. Then g is not the amalgamation of any set of functions $\{0, 1\} \to \{0\}$. However, if we use the *nonempty powerset monad* $\mathcal{P}^+$ (which corresponds to taking the colimit over the opposite of the category $\mathbf{Surj}^+$ of nonempty sets and surjections), then the amalgamation function gives us a surjection

$$\mathcal{P}^+([A, B]) \twoheadrightarrow [A, \mathcal{P}^+(B)]\,.$$

This means that we can recover the Kleisli category for the nonempty powerset monad on **Set** by taking the quotient by a suitable equivalence relation to the category $\mathbf{Set}/(\mathbf{Surj}^{+\,op})$.

In fact, this surjection has a natural right inverse, formed by sending a function $h :\ A \to \mathcal{P}^+(B)$ to the *largest* possible set $\mathcal{F} \subseteq [A, B]$ such that for all $f \in \mathcal{F}$ and $a \in A$, $f(a) \in h(a)$. The composition of these inverses sends a set of functions $A \to B$ to its ‘amalgamation closure’:

$$\mathrm{ac}(\mathcal{F}) = \{g :\ A \to B :\ \forall a \in A\,.\, g(a) = f(a) \text{ for some } f \in \mathcal{F}\}\,.$$

By restricting our set of morphisms, then, to those that are fixed points of this operator, we may recover the Kleisli category for $\mathcal{P}^+$.

In the next section, we will see how to generalize this idea to other monads, and to other categories.

1.6 Weakly filtered colimits

Suppose we have a lax action of a monoidal category $\mathcal{C}'$ upon a cocomplete category $\mathcal{C}$. Then we may copy the collapse from parametric promonads to promonads and define a monad on $\mathcal{C}$ given by:

$$MA = \varinjlim_{X\,:\,\mathcal{C}'} A.X\,.$$

Unfortunately, as in the case of the powerset monad above, this transformation does not commute with the collapse from parametric promonads to promonads. Indeed, if we consider our action as a parametric promonad and apply the collapse, then we get a category where the morphisms from A to B are given by the colimit

$$\varinjlim_{X\,:\,\mathcal{C}'} \mathcal{C}(A, X.B)\,,$$

while the morphisms from A to B in the Kleisli category for the monad M are given by

$$\mathcal{C}(A, \varinjlim_{X\,:\,\mathcal{C}'} X.B)\,.$$

There is a natural morphism from the first colimit into the second expression, but it is not an isomorphism in general. If $\mathcal{C}$ is the category of sets, there is a well known result that states that if $\mathcal{C}'$ is a κ-filtered category and $|A| < \kappa$, then the colimit will commute with the functor $[A, \_]$, but this does not help us, since the category $\mathcal{C}'$ is not filtered in most of the cases we are interested in (e.g., the category $\mathbf{Surj}^{op}$ has not got morphisms coequalizing parallel pairs, since such maps cannot be chosen to be surjective in general).

Instead, we try for the next best thing, which is to show that the natural map $\varinjlim_{X\,:\,\mathcal{C}'} \mathcal{C}(A, X.B) \to \mathcal{C}(A, \varinjlim_{X\,:\,\mathcal{C}'} X.B)$ is a surjection if $\mathcal{C}'$ satisfies some weaker conditions.

We define (see, for example, [Lüth and Ghani(1997), Def. 16]) a *κ-weakly filtered category* to be one in which every *discrete* diagram of size $< \kappa$ admits a cocone; i.e., a non-empty category such that whenever $(X_\alpha)_{\alpha<\kappa}$ is a collection of objects we have some object Y and morphisms $X_\alpha \to Y$ for each α.

Example 1. The category $\mathbf{Surj}^{+\,op}$ is κ-weakly filtered for arbitrary κ, because if $(X_\alpha)_{\alpha<\kappa}$ is a collection of sets, then we can take their Cartesian product, and the projections will be surjective.

The category $\mathbf{Surj}^{op}$ is not weakly filtered, because if there is a surjection from an object X to the empty set, then there is no surjection from X to any other set.

The importance of κ-weakly filtered limits is shown by the following result.

Proposition 1. *If $\mathcal{C}'$ is a κ-weakly filtered category, $D : \mathcal{C}' \to \mathbf{Set}$ is a functor and A is a set with cardinality less than κ, then the natural function*

$$\varinjlim_{X\,:\,\mathcal{C}'} [A, D(X)] \to [A, \varinjlim_{X\,:\,\mathcal{C}'} D(X)]$$

is a surjection.

Proof. Given a function $h : A \to \varinjlim_{X : \mathcal{C}'} D(X)$, for each $a \in A$, choose some $X_a : \mathcal{C}'$ such that $h(a) \in D(X_a)$. Since $\mathcal{C}'$ is κ-weakly filtered, there is some object Y of $\mathcal{C}'$ that admits morphisms out of all of the X_a, and therefore h may be considered as a function $A \to D(Y)$. $\square$

By setting $D(X) = X.B$, this means that if we have an action of $\mathcal{C}'$ on **Set**, where the monoidal category $\mathcal{C}'$ is κ-weakly filtered for any κ, then we get a full functor $\mathbf{Set}/\mathcal{C}' \to \mathrm{Kl}_M\,\mathbf{Set}$.

Moreover, if the colimit satisfies additional properties, then the Y in the proof of Proposition 1 can be chosen in a functorial manner, so that we get a faithful right inverse to this full functor.

Example 2. If $f : A \to \varinjlim_{X : \mathbf{Surj}^{+\,op}}[X, B]$ is a function with $f(a) \in [X_a, B]$ for each a, then f is equivalent to a function

$$\tilde{f} : A \to \left[\prod_{a \in A} X_a, B\right]$$

via the surjections $\prod_{a \in A} X_a \to X_b$ for each $b \in A$. The operator $f \mapsto \tilde{f}$ is well-defined in this case, and gives rise to the amalgamation functor that we considered in the previous section.

The point of these last two sections can be summarized as follows. If a monad M on **Set** can be expressed as the colimit of an action of a weakly filtered monoidal category $\mathcal{C}'$ on **Set**, then we do not lose very much by considering the category $\mathbf{Set}/\mathcal{C}'$ obtained from the action rather than the Kleisli category $\mathrm{Kl}_M\,\mathbf{Set}$. More specifically, there is a full functor $\mathrm{Kl}_M\,\mathbf{Set} \to \mathbf{Set}/\mathcal{C}'$ that is the identity on objects, so we may recover the Kleisli category by applying a suitable equivalence relation to the set of morphisms.

The benefit of reasoning about the action of $\mathcal{C}'$ rather than the monad M is that the action might be realizable in a much larger class of categories. For instance, the action of $\mathbf{Surj}^{+\,op}$ on **Set** may be replicated inside any monoidal closed category that admits a functor out of the category of sets (for example, most models of PCF or Idealized Algol).

1.7 Plan for the paper

The rest of this paper will consist in two sections. In the first, we will give a more detailed account of the construction that allows us to build a promonad (i.e., a category) out of a parametric monad.

In the second section, we will give a fully abstract game semantics for the language Probabilistic Algol (PA). This is by no means a new result – the paper [**?**] that introduced PA also gave it a fully abstract game semantics. Instead, the point of this section will be to demonstrate how the ideas we have outlined can be applied systematically to yield fully abstract models of programming languages.

2 A Kleisli-style construction for parametric monads

2.1 Parametric monads

Let $\mathcal{C}'$ be a monoidal category and let $\mathcal{C}$ be a category. Then a *lax left action* of $\mathcal{C}'$ on $\mathcal{C}$ is a functor $\_.\_ : \mathcal{C}' \times \mathcal{C} \to \mathcal{C}$ that gives rise (through currying) to a lax monoidal functor $\mathcal{C}' \to \mathrm{End}[\mathcal{C}, \mathcal{C}]$. In other words, we have natural transformations $\mathtt{passoc}_{X,Y,A} : X.Y.A \to (X \otimes Y).A$ and $\mathtt{1}_A : A \to I.A$ making the following diagrams commute for all objects A of $\mathcal{C}$ and X, Y, Z of $\mathcal{C}'$.

$$\begin{tikzcd}
X.Y.Z.A \arrow[r, "\mathtt{passoc}_{X,Y,Z.A}"] \arrow[dr, "X.\mathtt{passoc}_{Y,Z,A}"'] & (X \otimes Y).Z.A \arrow[r, "\mathtt{passoc}_{X\otimes Y,Z,A}"] & ((X \otimes Y) \otimes Z).A \arrow[d, "\mathtt{assoc}_{X,Y,Z}.A"] \\
& X.(Y \otimes Z).A \arrow[r, "\mathtt{passoc}_{X,Y\otimes Z,A}"] & (X \otimes (Y \otimes Z)).A
\end{tikzcd}$$

$$\begin{tikzcd}
X.A \arrow[r, "\mathtt{1}_{X.A}"] \arrow[dr, "\mathtt{lunit}_X.A"'] & I.X.A \arrow[d, "\mathtt{passoc}_{I,X,A}"] \\
& (I \otimes X).A
\end{tikzcd}
\qquad
\begin{tikzcd}
X.A \arrow[r, "X.\mathtt{1}_A"] \arrow[dr, "\mathtt{runit}_X.A"'] & X.I.A \arrow[d, "\mathtt{passoc}_{X,I,A}"] \\
& (X \otimes I).A
\end{tikzcd}$$

Example 3. – Any monad M is an action of the trivial category on its underlying category $\mathcal{C}$, regarding MA as the action $I.A$. The natural transformation $\mathtt{passoc}_{A,I,I}$ is precisely the monad action on A:

$$I.I.A = MMA \to MA = I.A = (I \otimes I).A\,.$$

Dually, any lax action gives rise to a monad on the category being acted upon, by setting $MA = A.I$.

– If $\mathcal{C}$ is also a monoidal category and $J : \mathcal{C}' \to \mathcal{C}$ is a lax monoidal functor with monoidal coherence μ, then there is an action of $\mathcal{C}'^{co}$ (i.e., $\mathcal{C}'$ with the opposite monoidal product) on $\mathcal{C}$ given by

$$A.X = A \otimes JX\,,$$

and

$$\mathtt{passoc}_{X,Y,A} = (A \otimes JY) \otimes JX \xrightarrow{\mathtt{assoc}_{A,JX,JY}} A \otimes (JY \otimes JX) \xrightarrow{A \otimes \mu_{X,Y}} A \otimes J(Y \otimes X)\,.$$

We sometimes refer to a left action of the opposite category $\mathcal{C}'^{co}$ on $\mathcal{C}$ as a *right action* of $\mathcal{C}'$ on $\mathcal{C}$. In that case, we write $X.A$ instead of $A.X$, so that the coherences become

$$\mathtt{passoc}_{A,X,Y} : A.X.Y \to A.(X \otimes Y)$$
$$\mathtt{r}_A : A \to A.I\,.$$

- If $\mathcal{C}$ is a monoidal closed category, and j an oplax monoidal functor with coherence ν, then there is an action of $\mathcal{C}'$ on $\mathcal{C}$ given by
$$A.X = jX \multimap A$$
and
$$\mathtt{passoc}_{A,X,Y} = jY \multimap (jX \multimap A) \to (jY \otimes jX) \multimap A \xrightarrow{\nu_{Y,X} \multimap A} j(Y \otimes X) \multimap A\,.$$
- The intersection of the first two examples is the *writer monad* given by $M_W X = X \otimes W$ for any monoid W in $\mathcal{C}$. The intersection of the first and third examples is the *reader monad* given by $M^R X = R \multimap X$ for any comonoid R in $\mathcal{C}$ (and, in particular, for any object R if $\mathcal{C}$ if the monoidal product in $\mathcal{C}$ is Cartesian).
  We shall therefore call an action of the form $A \otimes JX$ a *writer-style action* and one of the form $JX \multimap A$ a *reader-style action.*
- We can define an *oplax action* of $\mathcal{C}'$ on $\mathcal{C}$ to be a lax action of $\mathcal{C}'$ upon the opposite category $\mathcal{C}^{op}$. In this case, the coherence `passoc` goes from $A.(X \otimes Y)$ to $A.X.Y$. As monads are lax actions of the trivial category, so are comonads oplax actions of the trivial category.

References

[Ghica(2005)] Ghica DR (2005) Slot games: A quantitative model of computation. In: Proceedings of the 32Nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM, New York, NY, USA, POPL '05, pp 85–97, DOI 10.1145/1040305.1040313, URL http://doi.acm.org/10.1145/1040305.1040313

[Katsumata(2016)] Katsumata Sy (2016) Graded monads and semantics of effect systems, URL http://csl16.lif.univ-mrs.fr/planning/qslc/talks/katsumata/, qSLC 2016

[Lüth and Ghani(1997)] Lüth C, Ghani N (1997) Monads and modular term rewriting. In: Moggi E, Rosolini G (eds) Category Theory and Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 69–86

[Melliès(2012)] Melliès PA (2012) Parametric monads and enriched adjunctions. Preprint on webpage at https://www.irif.fr/~mellies/tensorial-logic/8-parametric-monads-and-enriched-adjunctions.pdf

[Moggi(1989)] Moggi E (1989) Computational lambda-calculus and monads. In: [1989] Proceedings. Fourth Annual Symposium on Logic in Computer Science, pp 14–23, DOI 10.1109/LICS.1989.39155

[Smirnov(2008)] Smirnov AL (2008) Graded monads and rings of polynomials. Journal of Mathematical Sciences 151(3):3032–3051, DOI 10.1007/s10958-008-9013-7, URL https://doi.org/10.1007/s10958-008-9013-7

[Spivey(1990)] Spivey M (1990) A functional theory of exceptions. Science of Computer Programming 14(1):25 – 42, DOI https://doi.org/10.1016/0167-6423(90)90056-J, URL http://www.sciencedirect.com/science/article/pii/016764239090056J

[Wadler(1990)] Wadler P (1990) Comprehending monads. In: Proceedings of the 1990 ACM Conference on LISP and Functional Programming, ACM, New York, NY, USA, LFP '90, pp 61–78, DOI 10.1145/91556.91592, URL http://doi.acm.org/10.1145/91556.91592

[Wadler(1992)] Wadler P (1992) The essence of functional programming. In: Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM, New York, NY, USA, POPL '92, pp 1–14, DOI 10.1145/143165.143169, URL http://doi.acm.org/10.1145/143165.143169