Parametric distributive laws

Uniform monad composition

Lorenzo Perticone

April 16, 2025

Department of Computer Science and Engineering,
Chalmers University of Technology and University of Gothenburg,
SE-41296 Gothenburg, Sweden.

Table of contents

1. Introduction
2. 2-Categories
3. Lax functor classifiers
4. Gray tensor product
5. Parametric distributive laws
6. Examples
7. Monad morphisms and distributive laws

Introduction

The problem: composing monads

Both computer scientists and mathematicians are interested in composing monads.

1. In universal algebra, algebraic topology/geometry, logic: composing 2 fixed monads
2. For programming language semantics: composing a monad with a family of (other) monads

The mathematicians' solution: distributive laws

Given 2 monads $\mathsf{T} = (T, \eta^T, \mu^T)$, $\mathsf{S} = (S, \eta^S, \mu^S)$ over the category $\mathcal{C}$

Definition (J. Beck, *Distributive Laws*, 1966)
A distributive law $\gamma : \mathsf{S} \rightsquigarrow \mathsf{T}$ is a natural transformation $\gamma : T \circ S \to S \circ T$ satisfying appropriate axioms.

This notion is extremely well behaved, allowing one to relate the Eilenberg-Moore categories of the two monads with that of the "composite".

An analogous notion can be obtained by swapping $\mathcal{T}$ and $\mathcal{S}$.

The computer scientists' solution: monad transformers

Given a monad $\mathsf{T} = (T, \eta, \mu)$ over a category $\mathcal{C}$

Definition (S. Liang, P. Hudak, M. Jones, *Monad transformers and modular interpreters*, 1995)
A monad transformer $\mathcal{T}_{\mathsf{T}} = (\mathcal{T}_T, \phi)$ for it is a pointed endofunctor on the category $\mathsf{Mnd}(\mathcal{C})$, such that $\mathcal{T}_T(\mathsf{Id}) = \mathsf{T}$

The computer scientists' solution: monad transformers

Given a monad $\mathsf{T} = (T, \eta, \mu)$ over a category $\mathcal{C}$

Definition (S. Liang, P. Hudak, M. Jones, *Monad transformers and modular interpreters*, 1995)
A monad transformer $\mathcal{T}_{\mathsf{T}} = (\mathcal{T}_T, \phi)$ for it is a pointed endofunctor on the category $\mathsf{Mnd}(\mathcal{C})$, such that $\mathcal{T}_T(\mathsf{Id}) = \mathsf{T}$

I have issues with this notion, the most readily seen being that it doesn't relate the multiplication of T with that of the resulting monad.

Outline

I want to suggest an alternate, more structured (but also more restrictive) approach to encompass the two notions. This will require some work:

1. Generalize our context from monads *over a category* to monads *in a 2-category*,
2. Recall (and slightly generalize) some well-known results from the theory of 2-categories,
3. Perform a couple of relatively simple calculations.

The first step alone will give us some flavor of the construction already, but it will become more usable later on. The rest of this presentation will consist of exploring what the construction looks like, and providing two examples.

2-Categories

Basics

A 2-category is made of objects, morphisms and 2-cells. Given 2 objects we get a hom-*category*, and composition of 1-cells is only unital and associative up to iso. They come in strict and non-strict versions.

Basics

A 2-category is made of objects, morphisms and 2-cells. Given 2 objects we get a hom-*category*, and composition of 1-cells is only unital and associative up to iso. They come in strict and non-strict versions.

Given 2-categories we can talk about 2-functors; they come in *strict, pseudo, lax and colax* versions.

Basics

A 2-category is made of objects, morphisms and 2-cells. Given 2 objects we get a hom-*category*, and composition of 1-cells is only unital and associative up to iso. They come in strict and non-strict versions.

Given 2-categories we can talk about 2-functors; they come in *strict, pseudo, lax and colax* versions.

There's also notions of (strict, pseudo, lax, colax) 2-natural transformations and only one notion of modifications, which will only play a small (but not unimportant) role in the following.

Functor 2-categories

As one can imagine functors, natural transformations and modification can be packaged in *2-functor 2-categories*. There's a zoo of such, but we'll only be interested in:

1. $\mathsf{St}[\mathcal{C}, \mathcal{D}]$ contains strict functors, strict transformations and modifications,
2. $\mathsf{St}_{\mathsf{Lax}}[\mathcal{C}, \mathcal{D}]$ contains strict functors, lax transformations and modifications,
3. $\mathsf{Lax}[\mathcal{C}, \mathcal{D}]$ contains lax functors, lax transformations and modifications.

We'll also breifly talk about $\mathsf{Ps}[\mathcal{C}, \mathcal{D}]$, but it won't play an important role.

Monads in 2-categories

Just as we do for the 2-category **Cat** of categories, we can talk about monads in a general 2-category $\mathcal{C}$

Definition (J. Bénabou, *Introduction to bicategories*, 1967)
A monad (a, t, η, μ) is given by an object $a : \mathcal{C}$, and endomorphism $\mathcal{T} : a \to a$ and 2-cells $\eta : id_a \to t$, $\mu : t \cdot t \to t$, subject to the usual axioms (after inserting the appropriate coherences)

As expected, in the case $\mathcal{C} = \mathbf{Cat}$, we obtain the usual notion of monads.

Monads and functors

In this general framework, something nice happens: lax (and hence pseudo and strict) 2-functors preserve monads. More is true:

Theorem
Monads in a 2-category $\mathcal{C}$ are 1-1 with lax functors $* \to \mathcal{C}$

Which leads us to define $\mathsf{Mnd}(\mathcal{C}) := \mathsf{Lax}[*, \mathcal{C}]$. The usual monad morphisms are morphisms in this 2-category, but there's more general morphisms (since we don't fix the underlying 2-cells).

Distributive laws are monads, too!

Our new, more general framework can start to pay us off:

Theorem
Distributive laws between monads in the 2-category $\mathcal{C}$ are just monads in the 2-category $\mathsf{Mnd}(\mathcal{C})$.

We can hence define $\mathsf{Dist}(\mathcal{C}) := \mathsf{Mnd}(\mathsf{Mnd}(\mathcal{C}))$.

Distributive laws are monads, too!

Our new, more general framework can start to pay us off:

Theorem
Distributive laws between monads in the 2-category $\mathcal{C}$ are just monads in the 2-category $\mathsf{Mnd}(\mathcal{C})$.

We can hence define $\mathsf{Dist}(\mathcal{C}) := \mathsf{Mnd}(\mathsf{Mnd}(\mathcal{C}))$.

It would be extremely neat if we could “curry” here:

$$\mathsf{Dist}(\mathcal{C}) = \mathsf{Lax}[*, \mathsf{Lax}[*, \mathcal{C}]] \simeq \mathsf{Lax}[* \boxtimes *, \mathcal{C}]$$

Distributive laws are monads, too!

Our new, more general framework can start to pay us off:

Theorem
Distributive laws between monads in the 2-category $\mathcal{C}$ are just monads in the 2-category $\mathsf{Mnd}(\mathcal{C})$.

We can hence define $\mathsf{Dist}(\mathcal{C}) := \mathsf{Mnd}(\mathsf{Mnd}(\mathcal{C}))$.

It would be extremely neat if we could "curry" here:

$$\mathsf{Dist}(\mathcal{C}) = \mathsf{Lax}[*, \mathsf{Lax}[*, \mathcal{C}]] \simeq \mathsf{Lax}[* \boxtimes *, \mathcal{C}]$$

Sadly, there's no $\boxtimes$ that allows for this (as far as I'm aware)!

Important tool (1): lax functor classifiers

The properties we want

We could make our previous calculation work if only we had a way to replace $*$ with some other 2-category $\hat{*}$, such that

$$\mathsf{Lax}[*, \mathcal{C}] \simeq \mathsf{St}_{\mathsf{Lax}}[\hat{*}, \mathcal{C}]$$

We could give a more formal description of what we want, using the language of 3-categories. But who's got time for that?

The properties we want

We could make our previous calculation work if only we had a way to replace $*$ with some other 2-category $\hat{*}$, such that

$$\mathsf{Lax}[*, \mathcal{C}] \simeq \mathsf{St}_{\mathsf{Lax}}[\hat{*}, \mathcal{C}]$$

We could give a more formal description of what we want, using the language of 3-categories. But who's got time for that?

The point is, this can be done. And the construction is not even *that* bad!

A decategorified example

There's a 2-monad over Cat whose pseudo algebras are monoidal categories. The induced forgetful functor $\mathsf{Alg}^{st} \to \mathsf{Alg}^{lax}$ has a left (2-)adjoint. How does this work?

A decategorified example

There's a 2-monad over **Cat** whose pseudo algebras are monoidal categories. The induced forgetful functor $\mathbf{Alg}^{st} \to \mathbf{Alg}^{lax}$ has a left (2-)adjoint. How does this work?

The objects of the new monoidal category are sequences of such, and morphisms are either sequences, or of the form $[A_1, \ldots, A_n] \to A_1 \otimes \ldots A_n$. We do this "operadically".

We then quotient the new morphisms in (the only) reasonable way.

This construction is called the "Lax morphism classifier" by R. Blackwell, G. M. Kelly, A. J. Power in *Two-dimensional monad theory* (1989).

The actual construction (sketch)

We can do something similar for 2-categories!

Theorem
Lax Functor Classifier Given a 2-category $\mathcal{C}$ there's a 2-category $\hat{\mathcal{C}}$ such that for every 2-category $\mathcal{D}$,

$$\mathsf{Lax}[\mathcal{C}, \mathcal{D}] \simeq \mathsf{St}_{\mathsf{Lax}}[\hat{\mathcal{C}}, \mathcal{D}]$$

The construction is almost identical as for monoidal categories, but objects are now the same and we play the game from the previous slide for *morphisms and 2-cells* (with the added constraint that they have to be composable).

The actual construction (sketch)

We can do something similar for 2-categories!

Theorem

Lax Functor Classifier Given a 2-category $\mathcal{C}$ there's a 2-category $\hat{\mathcal{C}}$ such that for every 2-category $\mathcal{D}$,

$$\mathsf{Lax}[\mathcal{C}, \mathcal{D}] \simeq \mathsf{St}_{\mathsf{Lax}}[\hat{\mathcal{C}}, \mathcal{D}]$$

The construction is almost identical as for monoidal categories, but objects are now the same and we play the game from the previous slide for *morphisms and 2-cells* (with the added constraint that they have to be composable).

We can also prove that this commutes with delooping monoidal categories! This implies, together with a small calculation, that

$$\hat{*} \simeq \widehat{\mathbb{B}*} \simeq \mathbb{B}\hat{*} \simeq \mathbb{B}\Delta_a$$

Important tool (2): the Gray tensor product

The universal property

The missing ingredient would be a left (2-) adjoint to the $\mathsf{St}_{\mathsf{Lax}}$ 2-functor.

The universal property

The missing ingredient would be a left (2-) adjoint to the $\mathsf{St}_{\mathsf{Lax}}$ 2-functor.

We don't know of any 2-adjoint, but there is an ordinary left adjoint in the literature: (the lax variant of) the Gray tensor product $\otimes_\ell$. It enjoys the following universal property

$$\mathsf{2Cat}[\mathcal{B}, St_{\mathsf{Lax}}[\mathcal{C}, \mathcal{D}]] \simeq \mathsf{2Cat}[\mathcal{B} \otimes_\ell \mathcal{C}, \mathcal{D}]$$

Where **2Cat** is the *category* of 2-categories. This was first introduced by J. W. Gray in *Formal category theory: adjointness for 2-categories* (1974).

The actual construction (sketch)

Constructing $\mathcal{C} \otimes_\ell \mathcal{D}$ goes something like this:

The actual construction (sketch)

Constructing $\mathcal{C} \otimes_\ell \mathcal{D}$ goes something like this:

1. The objects $a : \mathcal{C} \otimes_\ell \mathcal{D}$ are pairs $a = (c, d)$, with $c : \mathcal{C}$ and $d : \mathcal{D}$,

The actual construction (sketch)

Constructing $\mathcal{C} \otimes_\ell \mathcal{D}$ goes something like this:

1. The objects $a : \mathcal{C} \otimes_\ell \mathcal{D}$ are pairs $a = (c, d)$, with $c : \mathcal{C}$ and $d : \mathcal{D}$,
2. Morphisms generated (under composition) by pairs $f = (g, h)$, where one of the two is an identity (appropriately quotiented)

The actual construction (sketch)

Constructing $\mathcal{C} \otimes_\ell \mathcal{D}$ goes something like this:

1. The objects $a : \mathcal{C} \otimes_\ell \mathcal{D}$ are pairs $a = (c, d)$, with $c : \mathcal{C}$ and $d : \mathcal{D}$,
2. Morphisms generated (under composition) by pairs $f = (g, h)$, where one of the two is an identity (appropriately quotiented)
3. 2-cells are generated (under horiontal and vertical composition) by pairs of 2-cells (where one is the identity over the identity), plus "swaps":

$$\gamma_{g,h} : (g, id) \circ (id, h) \to (id, h) \circ (g, id)$$

quotiented by the appropriate relations (relating horizontal and vertical composites), plus relations encoding "naturality" for γ's.

The payoff: parametric n-fold monads

A simple calculation

We are now ready to harvest the full payoff of all this abstract nonsense. The first step is to compute define, for $\mathcal{C}, \mathcal{D}$ 2-categories

Definition
Parametric (iterated) monads The *set* of $\mathcal{D}$-parametric monads in $\mathcal{C}$ is

$$\begin{aligned}\mathsf{PMnd}(\mathcal{D},\mathcal{C}) :=& \mathsf{2Cat}[\mathcal{D}, \mathsf{Mnd}(\mathcal{C})] \\ =& \mathsf{2Cat}[\mathcal{D}, \mathsf{Lax}[*, \mathcal{C}]] \\ \simeq& \mathsf{2Cat}[\mathcal{D}, \mathsf{St}_{\mathsf{Lax}}[\mathbb{B}\Delta_a, \mathcal{C}]] \\ \simeq& \mathsf{2Cat}[\mathcal{D} \otimes_\ell \mathbb{B}\Delta_a, \mathcal{C}]\end{aligned}$$

A similar definition works for $\mathcal{D}$-parametric *n-fold* monads $\mathsf{PMnd}^n(\mathcal{D},\mathcal{C})$. We call the special case $n = 2$ *parametric distributive laws*, $\mathsf{PDist}(\mathcal{D},\mathcal{C})$.

Walking (semi-strict) gadgets!

It's worth taking a step back, and look at the simplest case for the 2-category of parameters: $\mathcal{D} = *$. Clearly, $* \otimes_\ell \mathbb{B}\Delta_a = \mathbb{B}\Delta_a$: this justifies calling it the *walking monad*.

[1]One might think we have to choose a way of associating $\otimes_\ell$, but carrying out the previous calculation shows that we *have* to associate to the left.

Walking (semi-strict) gadgets!

It's worth taking a step back, and look at the simplest case for the 2-category of parameters: $\mathcal{D} = *$. Clearly, $* \otimes_\ell \mathbb{B}\Delta_a = \mathbb{B}\Delta_a$: this justifies calling it the *walking monad*.

A similar argument suggests we call $(\mathbb{B}\Delta_a)^{\otimes_\ell 2}$ the *walking distributive law* and $(\mathbb{B}\Delta_a)^{\otimes_\ell n}$ the *walking n-fold monad*[1].

[1] One might think we have to choose a way of associating $\otimes_\ell$, but carrying out the previous calculation shows that we *have* to associate to the left.

What’s inside? (Or, Yang-Baxter for free!)

It is now worth looking at these “walking gadgets” more closely, to understand where each piece of data comes from.

What's inside? (Or, Yang-Baxter for free!)

It is now worth looking at these "walking gadgets" more closely, to understand where each piece of data comes from.

For walking distributive laws, the "actual distributive law" is given by the swaps that the Gray tensor product introduces.

What's inside? (Or, Yang-Baxter for free!)

It is now worth looking at these “walking gadgets” more closely, to understand where each piece of data comes from.

For walking distributive laws, the “actual distributive law” is given by the swaps that the Gray tensor product introduces.

For walking 3-fold monads (or more generally, for $n \geq 3$), it is just as clear that the 3 distributive laws we expect come from the same source. But we can also notice that they are related by Yang-Baxter equations (as noticed by E. Cheng in *Iterated distributive laws*, 2007): these fall out of the naturality equations we imposed while performing the Gray tensor product!

Two examples

Let’s be concrete for a second

We managed to produce a very abstract tool for talking about monads and their “compositions”. We now turn to well-understood cases, to see how our new shiny tool applies.

We’ll focus on the `Writer` and `Exception` monads, since there’s relatively straightforward constructions to make them fit here.

The writer monad

The fact that the `Writer` monad "composes with everything" can be encoded in a parametric distributive law.

The writer monad

The fact that the `Writer` monad "composes with everything" can be encoded in a parametric distributive law.

1. Consider a cartesian closed category $\mathcal{C}$. We'll be working with the 2-category $\mathbb{B}\mathsf{Fun}_{\mathcal{C}}[\mathcal{C}, \mathcal{C}]$, whose morphisms are $\mathcal{C}$-enriched endofunctors.

The writer monad

The fact that the `Writer` monad "composes with everything" can be encoded in a parametric distributive law.

1. Consider a cartesian closed category $\mathcal{C}$. We'll be working with the 2-category $\mathbb{B}\mathsf{Fun}_{\mathcal{C}}[\mathcal{C}, \mathcal{C}]$, whose morphisms are $\mathcal{C}$-enriched endofunctors.
2. We then construct a $\mathsf{Mnd}(\mathbb{B}\mathsf{Fun}_{\mathcal{C}}[\mathcal{C}, \mathcal{C}])$-parametric distributive law on it, exploiting the fact that being $\mathbb{C}$-enriched for a monads means exactly that it lifts to the Eilenberg-Moore categories for writer monads (a.k.a. categories of M-object / M-modules, for every monoid object $M : \mathcal{C}$)

The either monad

The game we play for the `Exception` monad is similar (i.e. we construct a similar ambient 2-category; the only difference is we don't need self-enrichment). Fix a category $\mathcal{C}$ with finitary coproducts.

Here, the key fact is that the Eilenberg-Moore category for the `Exception` monad is equivalent to the coslice over the parameter. It's then a straightforward computation to show that every monad lifts to it (a key point is that every monad is pointed).

The either monad

The game we play for the `Exception` monad is similar (i.e. we construct a similar ambient 2-category; the only difference is we don't need self-enrichment). Fix a category $\mathcal{C}$ with finitary coproducts.

Here, the key fact is that the Eilenberg-Moore category for the `Exception` monad is equivalent to the coslice over the parameter. It's then a straightforward computation to show that every monad lifts to it (a key point is that every monad is pointed).

But morphisms of monads aren't pointed; we need to restrict to the ones that are. In the end, what we get is a $\mathsf{Mnd}_*(\mathbb{B}\mathsf{Fun}[\mathcal{C}, \mathcal{C}])$-parametric distributive law.

One last treat: monad morphisms and distributive laws

A surprisingly weird question

Suppose given a category $\mathcal{C}$ and:

1. Monads

$$\mathsf{T}_1 = (T_1, \eta_1^T, \mu_1^T)\ ,\ \mathsf{T}_2 = (T_2, \eta_2^T, \mu_2^T)\ ,\ \mathsf{S}_1 = (S_1, \eta_1^S, \mu_1^S)\ ,\ \mathsf{S}_2 = (S_2, \eta_2^S, \mu_2^S)$$

2. Monad morphisms

$$\mathsf{F}_1 = (F_1, \phi_1) : \mathsf{T}_1 \to \mathsf{T}_2\ ,\ \mathsf{F}_2 = (F_2, \phi_2) : \mathsf{S}_1 \to \mathsf{S}_2$$

3. Distributive laws

$$\gamma_1 : \mathsf{T}_1 \rightsquigarrow \mathsf{S}_1\ ,\ \gamma_2 : \mathsf{T}_2 \rightsquigarrow \mathsf{S}_2$$

When do F_1, F_2 induce a monad morphism between the composite monads?

Whose answer is now straightforward

The answer turns out to be: when $F_1 = F_2$!

Whose answer is now straightforward

The answer turns out to be: when $F_1 = F_2$!

In this case, we can actually assemble the data from the previous slide in a morphism in the 2-category $\mathsf{Mnd}(\mathsf{Mnd}(\mathbb{B}\mathsf{Fun}[\mathcal{C}, \mathcal{C}))$ as follows:

$$((F, \phi_1), \phi_2) : ((T_1, \eta_1^T, \mu_1^T), (S_1, \gamma_1), \eta_1^S, \mu_1^S) \to ((T_2, \eta_2^T, \mu_2^T), (S_2, \gamma_2), \eta_2^S, \mu_2^S)$$

which, as we have seen, is exactly what we need. A few similar results are similarly straightforward from this perspective.

Future directions

1. Implementing some monad library that uses our notion of parametric distributive laws instead of monad transformers,
2. Formally verifying (fragments of) this result,
3. Extending this work to structured monads.

Future directions

1. Implementing some monad library that uses our notion of parametric distributive laws instead of monad transformers,
2. Formally verifying (fragments of) this result,
3. Extending this work to structured monads.

Thanks for bearing with me this long!