Unifying graded and parameterised monads

Dominic Orchard
School of Computing, University of Kent

Philip Wadler
School of Informatics, University of Edinburgh

Harley Eades III
Department of Computer Science, Augusta University

Monads are a useful tool for structuring effectful features of computation such as state, non-determinism, and continuations. In the last decade, several generalisations of monads have been suggested which provide a more fine-grained model of effects by replacing the single type constructor of a monad with an indexed family of constructors. Most notably, *graded monads* (indexed by a monoid) model effect systems and *parameterised monads* (indexed by pairs of pre- and post-conditions) model program logics. This paper studies the relationship between these two generalisations of monads via a third generalisation. This third generalisation, which we call *category-graded monads*, arises by generalising a view of monads as a particular special case of *lax functors*. A category-graded monad provides a family of functors $\mathsf{T}\,f$ indexed by morphisms f of some other category. This allows certain compositions of effects to be ruled out (in the style of a program logic) as well as an abstract description of effects (in the style of an effect system). Using this as a basis, we show how graded and parameterised monads can be unified, studying their similarities and differences along the way.

1 Introduction

Ever since Moggi [1989, 1991], monads have become an important structure in programming and semantics, particularly for capturing computational effects. In this approach, an effectful computation yielding a value of type A is modelled by the type $\mathsf{M}A$ where M is an endofunctor with the structure of a monad, i.e., with two operations (and some axioms): *unit* (η) mapping values into pure computations (with no effects) and *multiplication* (μ) composing effects of two computations (one nested in the other):

$$\eta_A : A \to \mathsf{M}A \qquad \mu_A : \mathsf{M}\,\mathsf{M}A \to \mathsf{M}A$$

Various generalisations of monads replace the single endofunctor M with a family of endofunctors indexed by effect information. For example, Wadler and Thiemann [2003] proposed a generalisation of monads to model effect systems using a family of monads $(\mathsf{M}^a)_{a\in E}$ indexed over a *bounded semilattice* $(E, \sqcup, \emptyset, \sqsubseteq)$. Later, *graded monads* were proposed (Katsumata [2014], Orchard et al. [2014]), generalising this idea to a family of endofunctors $(\mathsf{G}\,e)_{e\in E}$ indexed by the elements of an ordered monoid $(E, \bullet, I, \sqsubseteq)$ with monad-like unit and multiplication operations (and axioms) mediated by the monoid structure:

$$\eta_A : A \to \mathsf{G}\,I\,A \qquad \mu_{e,f,A} : \mathsf{G}\,e\,\mathsf{G}\,f\,A \to \mathsf{G}\,(e \bullet f)\,A \qquad \mathsf{G}\,(e \sqsubseteq f)A : \mathsf{G}\,e A \to \mathsf{G}\,f A$$

A family of approximation maps $\mathsf{G}\,(e \sqsubseteq f)$ is derived from the ordering.

Another indexed generalisation is provided by *parameterised monads*, which comprise a family of endofunctors $(\mathsf{P}(I,J))_{I\in\mathbb{I}^{\mathsf{op}},J\in\mathbb{I}}$ indexed by pairs of objects drawn from a category $\mathbb{I}$ which provides information akin to pre- and post-conditions (Wadler [1994], Atkey [2009b]). Parameterised monads have unit and multiplication operations (satisfying analogous axioms to monads):

$$\eta_{I,A} : A \to \mathsf{P}(I,I)A \qquad \mu_{I,J,K,A} : \mathsf{P}(I,J)\,\mathsf{P}(J,K)A \to \mathsf{P}(I,K)A \qquad \mathsf{P}(f,g)A : P(I,J)A \to P(I',J')A$$

Sam Lindley and Max S. New (Eds.): Eighth Workshop on Mathematically Structured Functional Programming (MSFP 2020)
EPTCS 317, 2020, pp. 18–38, doi:10.4204/EPTCS.317.2

The family of maps $\mathsf{P}(f,g)A$ (for all $f : I' \to I, g : J \to J'$) provides a notion of approximation via pre-condition strengthening and post-condition weakening. For pure computations via η, the pre-condition is preserved as the post-condition. For composition via μ, the post-condition of the outer computation must match the pre-condition of the inner, yielding a computation indexed by the pre-condition of the outer and post-condition of the inner.

Graded and parameterised monads are used for various kinds of fine-grained effectful semantics, reasoning, and programming. For example, graded monads model effect systems (Katsumata [2014], Mycroft et al. [2016]), trace semantics (Milius et al. [2015]), and session types (Orchard and Yoshida [2016]). Parameterised monads are used to refine effectful semantics with Floyd-Hoare triples (Atkey [2009b]), information flow tracking (Stefan et al. [2011]), and session types (Pucella and Tov [2008], Imai et al. [2010]). In GHC/Haskell, graded and parameterised monads are provided by the `effect-monad` package,[1] leveraging GHC's advanced type system features particularly in the case of graded monads.

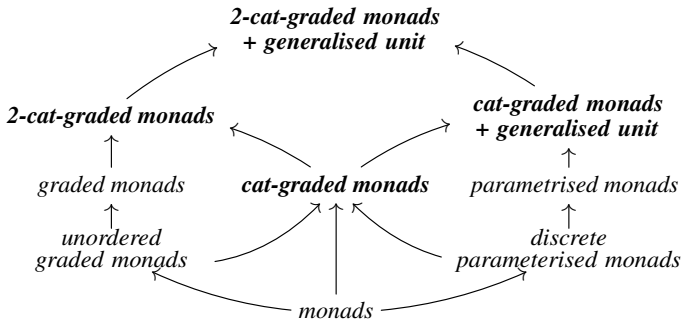
Both structures appear to follow a similar pattern, generalising monads into some indexed family of type constructors with monad-like operations mediated by structure on the indices. Furthermore, there are applications for which both graded and parameterised monads have been used, notably session types.[2] Thus, one might naturally wonder how graded and parameterised monads are related. We show that they can be related by a structure we call *category-graded monads*, based on a specialised class of *lax functors*.

Whilst graded monads are indexed by elements of a monoid and parameterised monads by pairs of indices, category-graded monads are indexed by the morphisms of a category, and have operations:

$$\eta_{I,A} : A \to \mathsf{T}\, id_I A \qquad \mu_{f,g,A} : \mathsf{T} f\, \mathsf{T} g A \to \mathsf{T}\,(g \circ f) A \qquad \mathsf{T}(\mathbf{f} : j \Rightarrow k)A : \mathsf{T}\, jA \to \mathsf{T}\, kA$$

for $f : I \to J$, $g : J \to K$, $j,k : I' \to J'$. Indexing by morphisms provides a way to restrict composition of effectful computations and a model that captures various kinds of indexing. The rightmost operation is provided by *2-category-graded monads* (generalising grading to 2-categories) where $\mathsf{T}\mathbf{f}$ lifts a 2-morphism $\mathbf{f}$ (morphism between morphisms), providing a notion of approximation akin to graded monads.

The structures we study along with their relationships are summarised by the diagram below, where the source of an arrow is at a more specific structure, and the target is a more general structure. Highlighted in bold are the new structures in this paper, centrally category-graded monads and its generalisation to *2-category-graded monads* and the additional structure of *generalised units*. Throughout, we state the results of the diagram below as propositions of the form "*every A is a B*" whenever there is an arrow from *A* to *B* in the diagram. By this we mean there is an injective map from *A* structures into *B* structures.



Section 3 introduces category-graded monads as a particular way of generalising monads via lax functors. Section 4 considers graded monads and 2-category-graded monads. Section 5 considers the notion of lax natural transformations used in Section 6 which considers parameterised monads and generalised units.

We show that the subclasses of *unordered* graded monads and *discrete* parameterised monads are both subsumed by category-graded monads. However, graded monads may have an ordering which provides approximation, requiring 2-category-graded monads, and full (non-discrete)

[1] https://hackage.haskell.org/package/effect-monad

[2] Orchard and Yoshida [2017] provide a survey of different structuring techniques for session types in the context of Haskell.

parameterised monads also have a kind of approximation which can be captured by a generalised unit for a category-graded monad. Section 7 shows that the apex of 2-category-graded monads plus generalised units capture analyses and semantics that have both graded and parameterised monad components, using an example of a Hoare logic for probabilistic computations.

2 Background

We first recall some standard, basic categorical facts relating monoids to categories which will be used throughout. We start by fixing some notation.

Notation 1 (2-Category)**.** For a 2-category $\mathbb{C}$, we denote the class of objects as $\mathbb{C}_0$, 1-morphisms as $\mathbb{C}_1$ and 2-morphisms as $\mathbb{C}_2$. We write 2-morphism arrows as $\Rightarrow$ unless they are natural transformations (2-morphisms in **Cat**) which are instead written as $\dot{\rightarrow}$. Appendix A provides a definition of 2-categories.

Notation 2 (Homset)**.** The *homset* of (1-)morphisms between any two objects $A, B \in \mathbb{C}_0$ is written $\mathbb{C}(A,B)$.

There are two standard ways to view monoids in categorical terms: Recall a set-theoretic presentation of a monoid with set M, operation $\bullet : M \times M \to M$ and unit $e \in M$. This is identical to a category $\mathbb{M}$, which has a single object $\mathbb{M}_0 = \{*\}$, morphisms as the elements of M, i.e., $\mathbb{M}_1 = M$, composition via the binary operation $\circ = \bullet$ and identity morphism $id_* = e$. Associativity and unit properties of categories and monoids align. Hence:

Proposition 3. Monoids are one-object categories.

An alternate but equivalent view of monoids is as a monoidal category, where $(M, \bullet, e)$ is a category with objects M, bifunctor $\bullet : M \times M \to M$ and unit object e. This monoidal category is *discrete*, meaning the only morphisms are the identities.

Proposition 4. Monoids are discrete monoidal categories.

In this paper, we study graded monads which are typically defined in the literature as being indexed by a pre-ordered monoid, or *pomonoid*. We therefore recall how the above two results generalise to pomonoids. A monoid $(M, \bullet, e)$ with a preorder $\sqsubseteq$ (with $\bullet$ monotonic wrt. $\sqsubseteq$)[3] is a 2-category following Proposition 3 but with added 2-morphisms between every pair of morphisms $x, y \in M$ whenever $x \sqsubseteq y$.

Proposition 5. Pre-ordered monoids are one-object 2-categories.

Pre-orders are categories, with a morphism for every pair of ordered elements. Thus we can replay Proposition 4, but we now have morphisms between some elements representing the ordering and thus the category is no longer discrete. However this monoidal category is *strict*, meaning that associativity and unit axioms are equalities rather than isomorphisms. (Note, discrete categories are automatically strict).

Proposition 6. Pre-ordered monoids are strict monoidal categories.

A further categorical view on monoids is that they can be identified as some distinguished object in a monoidal category:

Definition 7. A monoid in a monoidal category $(\mathbb{C}, \otimes, I)$ is a distinguished object $M \in \mathbb{C}$ equipped with a pair of morphisms $e : I \to M$ for the unit and $\bullet : M \otimes M \to M$, with associativity and unit axioms.

The *monoidal category of endofunctors* for some category is particularly important in this paper:

[3] That is for all $x_1, x_2, y_1, y_2 \in M$ then $x_1 \sqsubseteq y_1 \wedge x_2 \sqsubseteq y_2 \implies (x_1 \bullet x_2) \sqsubseteq (y_1 \bullet y_2)$

Definition 8. The category of endofunctors on $\mathbb{C}$, denoted $[\mathbb{C},\mathbb{C}]$, has endofunctors as objects and natural transformations as morphisms. This is a strict monoidal category with $([\mathbb{C},\mathbb{C}],\circ,\mathsf{Id}_{\mathbb{C}})$, i.e., with functor composition as the bifunctor and the identity endofunctor $\mathsf{Id}_{\mathbb{C}}A = A$ as the unit element.

The classic aphorism that *monads are just a monoid in the category of endofunctors* thus applies Definition 7 in the context of the monoidal category of endofunctors $([\mathbb{C},\mathbb{C}],\circ,\mathsf{Id}_{\mathbb{C}})$, pointing out that a monad for endofunctor $\mathsf{T} : \mathbb{C} \to \mathbb{C}$ is a particular single object. Since the tensor product of the monoidal category is $\circ$ then monad multiplication is the binary operator $\mu : \mathsf{T} \circ \mathsf{T} \dot{\to} T$ and the monad unit operation identifies the unit element $\eta : \mathsf{Id}_{\mathbb{C}} \dot{\to} T$. Thus:

Proposition 9. Monads are monoids in the category of endofunctors.

Finally, we leverage the (standard) equivalence between strict monoidal categories and 2-categories with one object, which "transposes" monoidal and 1-categorical composition into 1-categorical and 2-categorical composition respectively. That is, for a strict monoidal category $(\mathbb{C},\otimes,I)$ we can construct a one-object 2-category, call it $1(\mathbb{C})$, with $1(\mathbb{C})_0 = *$ and $1(\mathbb{C})_1 = \mathbb{C}_0$ and $1(\mathbb{C})_2 = \mathbb{C}_1$ with horizontal composition $\circ_0 = \otimes$ and identity morphism $id = I$, and vertical composition $\circ_1 = \circ_{\mathbb{C}}$ and 2-identities by identities of $\mathbb{C}$. Conversely, given a one-object 2-category, $\mathbb{C}$, we can construct a strict monoidal category $(\mathsf{SMC}(\mathbb{C}),\otimes,I)$ where $\mathsf{SMC}(\mathbb{C})_0 = \mathbb{C}_1$, $\mathsf{SMC}(\mathbb{C})_1 = \mathbb{C}_2$, $\otimes = \circ_0$, I is the 1-morphism identity, composition $\circ = \circ_1$ and identity is the 2-identity. The same result applies for discrete monoidal categories and one-object categories where we can elide the 2-categorical part. Hence:

Proposition 10. Discrete monoidal categories are equivalent to one-object categories, and strict monoidal categories are equivalent to one-object 2-categories.

Corollary 11. By Def. 8 and Prop. 10, the strict monoidal category of endofunctors $([\mathbb{C},\mathbb{C}],\circ,\mathsf{Id}_{\mathbb{C}})$ is equivalent to a one-object 2-category with single object $\mathbb{C}$, morphisms as endofunctors and 2-morphisms are natural transformations. For clarity, we denote this 2-category as **Endo**$(\mathbb{C})$.

3 Generalising monads via lax functors to category-graded monads

Recall that every monoid corresponds to a single-object category (Proposition 3). In a single-object category, all morphisms compose just as all elements of a monoid multiply. Categories can therefore be seen as a generalisation of monoids: morphisms f and g compose to $g \circ f$ only when the source of g agrees with the target of f. Similarly, *groupoids* generalise the notion of groups to categories, where elements of the group are morphisms and every morphism has an inverse.

This process of generalising some notion to a category is known as *horizontal categorification* or *oidification* (echoing the relationship of groups to groupoids) (nLab authors [2020], Bertozzini et al. [2008a,b]). The general approach is to realise some concept as a kind of category comprising a single object which can then be generalised to many objects; categories are the "oidification" of monoids going from a single object to many (nLab authors [2020])—though the term *category* is preferred to *monoidoid*!

Our approach can be summarised as the horizontal categorification of monads. It turns out that this yields structures that unify graded and parameterised monads, but with some subtleties as we shall see.

First we need to understand in what way the concept of a monad can be seen as single-object entity such that it can be subjected to oidification. The view of a monad as a monoid in category of endofunctors (Proposition 9) highlights that a monad comprises a single object in $[\mathbb{C},\mathbb{C}]$ but oidification cannot readily be applied to this perspective. Instead, we take Bénabou's [1967] view of monads as *lax functors* which is more amenable to oidification. We first recall the definitions of lax functors for completeness.

Definition 12. A *lax* functor $\mathsf{F} : \mathbb{C} \to \mathbb{D}$ (where $\mathbb{D}$ is a 2-category)[4] comprises an object mapping and a morphism mapping, however the usual functor axioms $id_{\mathsf{F}A} = \mathsf{F}id_A$ and $\mathsf{F}g \circ \mathsf{F}f = \mathsf{F}(g \circ f)$ are replaced by families of 2-morphisms in $\mathbb{D}$ which "laxly" preserve units and composition:

$$\eta_A : id_{\mathsf{F}A} \Rightarrow \mathsf{F}id_A \qquad \mu_{f,g} : \mathsf{F}g \circ \mathsf{F}f \Rightarrow \mathsf{F}(g \circ f)$$

We have chosen the names of these families to be suggestive of our endpoint here.

Whilst the axioms of a category are preserved automatically by non-lax functors, this is not the case here. For example, if F is a functor then $\mathsf{F}f = \mathsf{F}(id \circ f) = \mathsf{F}id \circ \mathsf{F}f = \mathsf{F}f$, but not if F is a lax functor. Instead a lax functor has additional axioms for associativity of μ and unitality of η:

$$\begin{array}{ccc} \mathsf{F}f & \xrightarrow{\eta_J \mathsf{F}f} & \mathsf{F}id_J \circ \mathsf{F}f \\ {\scriptstyle \mathsf{F}f\,\eta_I}\downarrow & & \downarrow{\scriptstyle \mu_{id_J,f}} \\ \mathsf{F}f \circ \mathsf{F}id_I & \xrightarrow[\mu_{f,id_I}]{} & \mathsf{F}f \end{array} \qquad \begin{array}{ccc} \mathsf{F}f \circ \mathsf{F}g \circ \mathsf{F}h & \xrightarrow{\mathsf{F}f\,\mu_{g,h}} & \mathsf{F}f \circ \mathsf{F}(g \circ h) \\ {\scriptstyle \mu_{f,g}\,\mathsf{F}h}\downarrow & & \downarrow{\scriptstyle \mu_{f,g\circ h}} \\ \mathsf{F}(f \circ g) \circ \mathsf{F}h & \xrightarrow[\mu_{f\circ g,h}]{} & \mathsf{F}(f \circ g \circ h) \end{array}$$

We can thus see monads as lax functors between the terminal category **1** and the one-object 2-category of endofunctors on $\mathbb{C}$:

Proposition 13 (Bénabou [1967])**.** For a category $\mathbb{C}$, a *monad* on $\mathbb{C}$ is a lax functor $\mathsf{T} : \mathbf{1} \to \mathbf{Endo}(\mathbb{C})$ where **1** is the single-object category with $* \in \mathbf{1}_0$ and a single morphism $id_* : * \to *$. Then $\mathsf{T}* = \mathbb{C}$ and $\mathsf{T}id_*$ identifies an endofunctor on $\mathbb{C}$. Laxness means the functor axioms for T are 2-morphisms, which are the unit and multiplication operations of the monad on the endofunctor $\mathsf{T}id_*$:

$$\eta : id_{\mathsf{T}*} \dot{\to} \mathsf{T}\,id_* \qquad \mu : \mathsf{T}\,id_* \mathsf{T}\,id_* \dot{\to} \mathsf{T}\,id_*$$

where $id_{\mathsf{T}*}$ is the identity endofunctor Id on $\mathbb{C}$. The monad axioms are exactly the unit and associativity axioms of the lax functor.

This proposition recasts the aphorism that *monads are monoids in the category of endofunctors*. It equivalently views monads as lax homomorphisms (lax functors) between the singleton monoid **1** and the monoid of endofunctors. Since the source category **1** has but one object and one morphism, T identifies a particular endofunctor on $\mathbb{C}$ and the lax operators η and μ provide the usual monad operations for T.

We can now "oidify" this definition in two ways: (1) generalising the singleton source **1** to a category, or (2) generalising the singleton target category $\mathbf{Endo}(\mathbb{C})$ to **Cat**. In this paper, we pursue the first choice, though we discuss the second in Section 8.3. We thus oidify Bénabou's monad definition by replacing the singleton category **1** with an arbitrary category $\mathbb{I}$. We might have chosen the name *monadoid* for this structure, but since there are two ways in which the oidification can be applied, we settle on the name *category-graded monads*, the terminology for which will be explained once we recall graded monads in the next section. We also found that people do not like the name *monadoid*.

Definition 14. Let $\mathbb{I}$ and $\mathbb{C}$ be categories. A *category-graded monad* is a lax functor $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$, and because $\mathbf{Endo}(\mathbb{C})$ has only a single object we require that $\mathsf{T}I = \mathbb{C}$ for all $I \in \mathbb{C}_0$. Thus the morphism mapping of T can be thought of as family of endofunctors indexed by $\mathbb{I}$ morphisms, i.e., for all $f : I \to J$ in $\mathbb{I}$ then $\mathsf{T}f : \mathbb{C} \to \mathbb{C}$ is an endofunctor.

We refer to $\mathbb{I}$ as the indexing category and often write $\mathsf{T}f$ as T_f. For brevity we sometimes refer to category-graded monads as *cat-graded monads* or $\mathbb{I}$*-graded monads* if the category $\mathbb{I}$ is in scope.

[4] *Lax functors* are often defined between 2-categories, but can instead be defined (like here) with a 1-category in the domain which is treated as a 2-category with only trivial 2-morphisms.

The definition of category-graded monads is compact, but the requirement that T is a lax functor comes with a lot of structure which is akin to that of monads and graded monads.

Corollary 15 (Category-graded monad operations and axioms)**.** Suppose $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ is a category-graded monad as defined above. Then, following from the lax functor definition for T there are natural transformations (which we call *unit* and *multiplication* respectively):

$$\eta_I : id_{\mathsf{T}I} \dot{\to} \mathsf{T}_{id_I} \qquad \mu_{f,g} : \mathsf{T}_f \mathsf{T}_g \dot{\to} \mathsf{T}_{(g \circ f)} \qquad (\textit{where } f : I \to J \text{ and } g : J \to K \text{ are in } \mathbb{I})$$

Following from the lax functor, these satisfy associativity and unitality axioms (specialised from Def. 12):

$$\begin{array}{ccc} \mathsf{T}_f & \xrightarrow{\eta_J \mathsf{T}_f} & \mathsf{T}_{id_J} \mathsf{T}_f \\ {\scriptstyle \mathsf{T}_f \eta_I}\downarrow & \searrow & \downarrow{\scriptstyle \mu_{id_J,f}} \\ \mathsf{T}_f \mathsf{T}_{id_I} & \xrightarrow[\mu_{f,id_I}]{} & \mathsf{T}_f \end{array} \qquad \begin{array}{ccc} \mathsf{T}_f \mathsf{T}_g \mathsf{T}_h & \xrightarrow{\mathsf{T}_f \mu_{g,h}} & \mathsf{T}_f \mathsf{T}_{(h \circ g)} \\ {\scriptstyle \mu_{f,g} \mathsf{T}_h}\downarrow & & \downarrow{\scriptstyle \mu_{f,h \circ g}} \\ \mathsf{T}_{(g \circ f)} \mathsf{T}_h & \xrightarrow[\mu_{g \circ f,h}]{} & \mathsf{T}_{h \circ g \circ f} \end{array}$$

Note that the left square uses the equality $\mathsf{T}_{(f \circ id_I)} = \mathsf{T}_{(id_J \circ f)} = \mathsf{T}_f$ due to the unitality of *id* and the right square uses associativity of composition such that $\mathsf{T}_{h \circ (g \circ f)} = \mathsf{T}_{(h \circ g) \circ f}$

Example 16. Following Prop. 13 and the above definition, every monad $(\mathsf{M} : \mathbb{C} \to \mathbb{C}, \mu, \eta)$ is a category-graded monad on $\mathbb{C}$, with indexing $\mathbb{I} = \mathbf{1}$, lax functor $\mathsf{T}_{id_*} = \mathsf{M}$, and operations $\mu_{id_*,id_*} = \mu$ and $\eta_* = \eta$.

Remark 17. For morphisms $f : I \to J, g : J \to K$ in $\mathbb{I}$, any two morphisms $j : A \to \mathsf{T}_f B$ and $k : B \to \mathsf{T}_g C$ in $\mathbb{C}$ can then be composed using multiplication:

$$A \xrightarrow{j} \mathsf{T}_f B \xrightarrow{\mathsf{T}_f k} \mathsf{T}_f \mathsf{T}_g C \xrightarrow{\mu_{f,g,C}} \mathsf{T}_{g \circ f} C \tag{1}$$

This composition is akin to Kleisli composition of a monad and has identities given by $\eta_{I,A}$. This shows the role of the opposite category in $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$: the outer functor in the source of μ (T_f of $\mathsf{T}_f \mathsf{T}_g$) corresponds to the first effect and the inner (T_g of $\mathsf{T}_f \mathsf{T}_g$) corresponds to the second effect. Sequential composition via $\mu_{f,g,C}$ then returns an object $\mathsf{T}_{g \circ f} C$.

Remark 18. The above composition for morphisms of the form $A \to \mathsf{T}_f B$ corresponds to the *Grothendieck construction* for an indexed category $\mathsf{T}' : \mathbb{I}^{\mathsf{op}} \to \mathbf{Cat}$ (Grothendieck [1961]) which maps an $\mathbb{I}^{\mathsf{op}}$-indexed category T' to a category $\mathbb{I} \int \mathsf{T}'$ which provides a category with:

- objects $(\mathbb{I} \int \mathsf{T}')_0 = \mathbb{I}_0 \times \mathbb{C}_0$ i.e., pairs of index and base category objects;
- morphisms $g : I \times A \to J \times B$ given by $g = f \times h$ where $f : I \to J \in \mathbb{I}_1$ and $h : A \to \mathsf{T}'_f B \in \mathbb{C}_1$;
- composition and identities defined as in Remark 17 in terms of the lax functor operations μ and η.

As pointed out by Jacobs [1999], μ and η need not be natural isomorphisms but just natural transformations in order for the above construction to be a category [Jacobs, 1999, p.117, 1.10.7]; that is, T' need only be a lax functor for the Grothendieck construction to work, as is the case for category-graded monads here.

Example 19. The *identity category-graded monad* is a lifting of the identity monad to an arbitrary indexing category. We denote this $\mathbf{Id} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ with $\mathbf{Id}_f = \mathsf{Id}_{\mathbb{C}}$. This will be useful later.

Example 20. A category can be viewed as a state machine: objects as states and morphisms as transitions. Given a monad M on $\mathbb{C}$ and some category $\mathbb{I}$, there is a category-graded monad $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ with $\mathsf{T}_f = \mathsf{M}$, restricting the composition of effectful computations by the morphisms of $\mathbb{I}$. Any effectful

operation producing values $\mathsf{M}A$ can be suitably wrapped into $\mathsf{T}_f A$ for some particular f describing the operation and its corresponding state transition. A cat-graded monad model for a program then provides a *static trace* of the effects in its indices. We may also wrap a monad into a category-graded monad but with some additional implementation related to the grading, with the implementation dependent on the indices.

For example, we could capture the simple state-machine protocol of a mutual exclusion lock over a memory cell with $\mathbb{I}_0 = \{\mathsf{free}, \mathsf{critical}\}$ and morphisms $\mathsf{lock} : \mathsf{free} \to \mathsf{critical}$ and $\mathsf{unlock} : \mathsf{critical} \to \mathsf{free}$ and $\mathsf{get}, \mathsf{put} : \mathsf{critical} \to \mathsf{critical}$. We can then wrap the state monad, call it St^S for some state type S, to get an $\mathbb{I}$-graded monad ConcSt^S, wrapping the usual state monad operations as $get : \mathsf{ConcSt}^S_{\mathsf{get}}\ S$ and $put : S \to \mathsf{ConcSt}^S_{\mathsf{put}}\ 1$ and adding operations (implemented in ConcSt) $lock : \mathsf{ConcSt}^S_{\mathsf{lock}}\ 1$ and $unlock : \mathsf{ConcSt}^S_{\mathsf{unlock}}\ 1$ and an operation for spawning threads from computations whose index is any morphism from free to free, i.e. $spawn : (\forall f.\mathsf{ConcSt}^S_{f:\mathsf{free}\to\mathsf{free}}\,1) \to \mathsf{ConcSt}^S_{id_{\mathsf{free}}}\,1$. Subsequently, the category grading statically ensures the mutual exclusion protocol: a spawned thread must acquire the *lock* before it can *get* or *put* (or indeed *unlock*), and the lock must be released if acquired. In Haskell-style syntax, we can then have programs like: $\mathbf{do}\{lock;\ x \leftarrow get;\ put(x+1);\ unlock\} : \mathsf{ConcSt}^{\mathsf{Int}}_{\mathsf{unlock}\circ\mathsf{put}\circ\mathsf{get}\circ\mathsf{lock}:\mathsf{free}\to\mathsf{free}}\ 1$.

4 Graded monads as category-graded monads

Graded monads are a generalisation of monads from a single endofunctor to an indexed family of endofunctors whose indices are drawn from a (possibly ordered) monoid. The graded monad operations are "mediated" by the monoidal structure of the indices. Graded monads appear in the literature under various names: *indexed monads* (Orchard et al. [2014]), *parametric effect monads* (Katsumata [2014]) or *parametric monads* (Mellies [2012]). The terminology of *graded monads* (e.g., in Smirnov [2008], Fujii et al. [2016], Milius et al. [2015], Gaboardi et al. [2016]) is now the preferred term in the community.

Here we show that lax functors $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ generalise the notion of graded monads. We thus explain why we chose the name *category-graded monads*, since this structure can be seen as generalising a monoid-based grading to a category-based grading. The idea of generalising graded monads to lax functors is mentioned by [Katsumata, 2014, §6.2] and Fujii et al. [2016]. We give the full details.

Definition 21. Let $(M, \bullet, e, \sqsubseteq)$ be a pomonoid, meaning that $\bullet$ is also monotonic with respect to $\sqsubseteq$, presented as a strict monoidal category (Prop. 6) which we denote as M for convenience.

A *graded monad* comprises a functor $\mathsf{G} : M \to [\mathbb{C}, \mathbb{C}]$ which is *lax monoidal* and therefore has natural transformations witnessing lax preservation of the monoidal structure of M into the monoidal structure of $[\mathbb{C}, \mathbb{C}]$, that is *unit* $\eta^{\mathsf{G}} : \mathsf{Id} \dot{\to} \mathsf{G}\,e$ and *multiplication* $\mu^{\mathsf{G}}_{m,n} : \mathsf{G}\,m\,\mathsf{G}\,n \dot{\to} \mathsf{G}\,(m \bullet n)$. The morphism mapping of the functor G means that an ordering $m \sqsubseteq n$ corresponding to a morphism $f : m \to n \in M_1$ is then mapped to a natural transformation $\mathsf{G}f : \mathsf{G}\,m \dot{\to} \mathsf{G}\,n$ which we call an *effect approximation*. Lax monoidality of G means that functoriality of $\bullet$ is laxly preserved by μ^{G}. A graded monad thus satisfies axioms:

$$\begin{array}{ccc} \mathsf{G}\,m & \xrightarrow{\mathsf{G}\eta} & \mathsf{G}\,m\mathsf{G}\,e \\ {\scriptstyle \eta\mathsf{G}}\downarrow & \searrow & \downarrow{\scriptstyle \mu_{m,e}} \\ \mathsf{G}\,e\mathsf{G}\,m & \xrightarrow[\mu_{e,m}]{} & \mathsf{G}\,m \end{array} \qquad \begin{array}{ccc} \mathsf{G}\,m\mathsf{G}\,n\mathsf{G}\,p & \xrightarrow{\mathsf{G}\mu_{n,p}} & \mathsf{G}\,m\mathsf{G}\,(n \bullet p) \\ {\scriptstyle \mu_{m,n}\mathsf{G}}\downarrow & & \downarrow{\scriptstyle \mu_{m,n\bullet p}} \\ \mathsf{G}\,(m \bullet n)\mathsf{G}\,p & \xrightarrow[\mu_{m\bullet n,p}]{} & \mathsf{G}\,(m \bullet n \bullet p) \end{array} \qquad \begin{array}{ccccc} \mathsf{G}n\mathsf{G}m & \xrightarrow{\mathsf{G}f\mathsf{G}} & \mathsf{G}n'\mathsf{G}m & \xrightarrow{\mathsf{G}\mathsf{G}g} & \mathsf{G}n'\mathsf{G}m' \\ {\scriptstyle \mu^{\mathsf{G}}_{n,m}}\downarrow & & & & \downarrow{\scriptstyle \mu^{\mathsf{G}}_{n',m'}} \\ \mathsf{G}(n \bullet m) & & \xrightarrow[\mathsf{G}(f \bullet g)]{} & & \mathsf{G}(n' \bullet m') \end{array}$$

where $f : n \to n', g : m \to m' \in M_1$ in the rightmost diagram.

Note, graded monads are not families of monads; $\mathsf{G}\,m$ need not be a monad for all m. For example:

Example 22. The *graded list monad* is indexed by the monoid $(\mathbb{N}, *, 1, \leq)$ refining the usual list monad by length $\mathsf{G}_n A = A^0 + A^1 + \ldots + A^n$ thus $\mathsf{G}_n A$ represents the type of lists of length at most n with elements of

type A. This graded monad then has the operations $\eta^{\mathsf{G}}_A : A \to \mathsf{G}_1 A = \mathsf{in}_1$ injecting a value into a singleton list and $\mu^{\mathsf{G}}_{n,m,A} : \mathsf{G}_m(\mathsf{G}_n A) \to \mathsf{G}_{m*n} A = \mathit{concat}$ which concatenates together a list of lists and approximation $\mathsf{G}(f : m \leq n) : \mathsf{G}_m A \to \mathsf{G}_n A$ which views a list of at most length m as a list of at most length n when $m \leq n$.

In the literature, graded monads need not have a pre-ordering (e.g., Mycroft et al. [2016], Gibbons [2016]), so we may distinguish two kinds of graded monad depending on the "grading" structure: *monoid-graded monads* and *pomonoid-graded monads*. A monoid-graded monad $\mathsf{G} : A \to [\mathbb{C}, \mathbb{C}]$ therefore maps from a *discrete* strict monoidal category, i.e., one whose only morphisms are the identities. Generally when we refer to "graded monads" we mean pomonoid-graded monads since this is the most common meaning in the literature. We qualify the nature of the grading structure when we need to be explicit.

We now come to the first main result: that monoid-graded monads are a simple case of category-graded monads via the old idea of monoids as single-object categories (Prop. 3) and monoidal categories (Prop. 4):

Proposition 23. Every monoid-graded monad is a category-graded monad.

Proof. The indexing category for a monoid-graded monad on $\mathbb{C}$ is discrete monoidal $(M, \bullet, e)$, thus it is equivalent to a single-object category (Prop. 10), which we write as $1(M)$. Similarly, $[\mathbb{C}, \mathbb{C}]$ is equivalent to the single object 2-category $\mathbf{Endo}(\mathbb{C})$ (Cor. 11). Thus $\mathsf{G} : M \to [\mathbb{C}, \mathbb{C}]$ is equivalent to a lax functor $\mathsf{T} : 1(M) \to \mathbf{Endo}(\mathbb{C})$, with $\mathsf{T} f = \mathsf{G} f$ (where f is a morphism of $1(M)$ and an object of M) and $\eta_1 = \eta^{\mathsf{G}}$ and $\mu_{f,g} = \mu^{\mathsf{G}}_{f,g}$ whose lax functor axioms follow directly from the graded monad axioms. □

On the naming The general paradigm of "grading" is to have an indexed structure where the structure of the indices matches the structure of some underlying semantics or term calculus (Orchard et al. [2019]). We can grade by different structures, mapping (e.g., as a homomorphism or functor) to the underlying domain (in our case $\mathbf{Endo}(\mathbb{C})$). Thus, we can view this particular class of lax functors as a horizontal categorification of monads which serves to "grade" by a category, rather than grading by a single object as with monads or grading by a (po)monoid as with traditional graded monads.

4.1 Pomonoid-graded monads and 2-category-graded monads

Proposition 23 considers only monoid-graded monads (without an ordering) however in the literature *pomonoid*-graded monads are the norm. Since strict monoidal categories are equivalent to one-object 2-categories (Prop. 10), where the morphisms of the former become the 2-morphisms of the latter, we therefore generalise category-graded monads to 2-category-graded monads to complete the picture.

Definition 24. A *2-category-graded monad* extends a category-graded monad to a 2-categorical index category, thus $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ is a lax 2-functor. This provides a 2-morphism mapping from $\mathbb{I}_2$ to natural transformations on $\mathbb{C}$: for all $f, g : A \to B \in \mathbb{I}_1$ and $\mathbf{f} : f \Rightarrow g \in \mathbb{I}_2$ then $\mathsf{T}_{\mathbf{f}} : \mathsf{T}_f \dot{\to} \mathsf{T}_g$ is a natural transformation which we call an *approximation* to recall the similar idea in graded monads.

The 2-morphism mapping has 2-functorial axioms in two flavours: for vertical composition (left two diagrams) and for horizontal composition which is laxly preserved by T (right two diagrams):

$$\begin{array}{ccc} \mathsf{T}_f & \xrightarrow{\mathbf{id}_{\mathsf{T}_f}} & \mathsf{T}_f \\ {\scriptstyle \mathsf{T}_{\mathbf{id}_f}}\downarrow & & \| \\ \mathsf{T}_f & = & \mathsf{T}_f \end{array} \qquad \begin{array}{ccc} \mathsf{T}_f & \xrightarrow{\mathsf{T}_{\mathbf{g} \circ_1 \mathbf{f}}} & \mathsf{T}_h \\ {\scriptstyle \mathsf{T}_{\mathbf{f}}}\downarrow & & \\ \mathsf{T}_g & \xrightarrow[\mathsf{T}_{\mathbf{g}}]{} & \end{array} \qquad \begin{array}{ccc} \mathsf{Id}_{\mathbb{C}} & \xrightarrow{\eta_I} & \\ {\scriptstyle \eta_I}\downarrow & & \\ \mathsf{T}_{id_I} & \xrightarrow[\mathsf{T}_{\mathbf{id}_{id_I}}]{} & \mathsf{T}_{id_I} \end{array} \qquad \begin{array}{ccc} \mathsf{T}_f \mathsf{T}_g & \xrightarrow{\mathsf{T}_{\mathbf{f}} \mathsf{T}_{\mathbf{g}}} & \mathsf{T}_{f'} \mathsf{T}_{g'} \\ {\scriptstyle \mu_{f,g}}\downarrow & & \downarrow{\scriptstyle \mu_{f',g'}} \\ \mathsf{T}_{g \circ f} & \xrightarrow[\mathsf{T}_{\mathbf{g} \circ_0 \mathbf{f}}]{} & \mathsf{T}_{g' \circ f'} \end{array}$$

The left two diagrams hold for all $f, g, h : A \to B \in \mathbb{I}_1$ and $\mathbf{f} : f \Rightarrow g \in \mathbb{I}_2$ and $\mathbf{g} : g \Rightarrow h \in \mathbb{I}_2$. The right two diagrams hold for all $f, f' : A \to B \in \mathbb{I}_1$ and $g, g' : B \to C \in \mathbb{I}_1$ and $\mathbf{f} : f \Rightarrow f' \in \mathbb{I}_2$ and $\mathbf{g} : g \Rightarrow g' \in \mathbb{I}_2$.

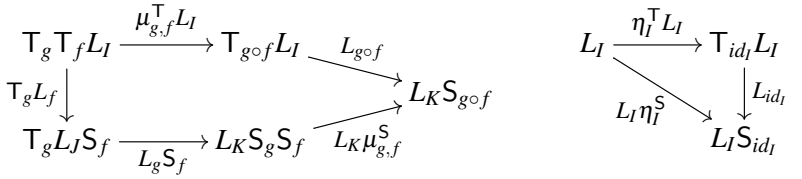
Proposition 25. Every pomonoid-graded monad is a 2-category-graded monad.

Proof. Strict monoidal categories are equivalent to single-object 2-categories (Prop. 10). Thus $\mathsf{G} : M \to [\mathbb{C},\mathbb{C}]$ is equivalent to a lax functor $\mathsf{T} : 1(M) \to \mathbf{Endo}(\mathbb{C})$, with $\mathsf{T}f = \mathsf{G}f$ (where f is a morphism of $1(M)$ and an object of M) and $\mathsf{T}\mathbf{f} = \mathsf{G}\mathbf{f}$ for approximations (where $\mathbf{f}$ is a 2-morphism of $1(M)$ and a morphism of M) and $\eta_1 = \eta^{\mathsf{G}}$ and $\mu_{f,g} = \mu^{\mathsf{G}}_{f,g}$ whose lax functor axioms follow from the graded monad axioms. □

5 Lax natural transformations and category-graded monad morphisms

Our use of lax functors provides a source of useful results from the literature. Here we show the notion of *lax natural transformations* (Street [1972]) which provides category-graded monad (homo)morphisms and further useful additional structure leveraged in Section 6.

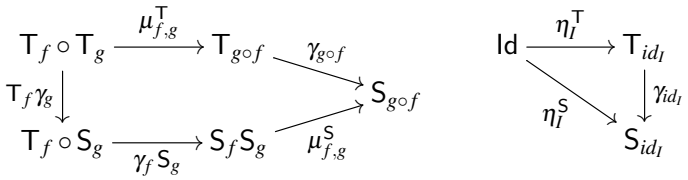
Definition 26. (Street [1972]) Let $\mathbb{I}$ be a category and $\mathsf{S},\mathsf{T} : \mathbb{I} \to \mathbf{Cat}$ be two lax functors. A *left lax natural transformation* $L : \mathsf{S} \dot{\to} \mathsf{T}$ comprises (1) a functor $L_I : \mathsf{S}I \to \mathsf{T}I$ for every $I \in \mathbb{I}_0$ and (2) a natural transformation $L_f : \mathsf{T}_f L_I \dot{\to} L_J \mathsf{S}_f$ for every $f : I \to J \in \mathbb{I}_1$, such that the following diagrams commute (where $f : I \to J$ and $g : J \to K$):



A *right lax natural transformation* $R : \mathsf{S} \dot{\to} \mathsf{T}$ has the same data, with functors $R_I : \mathsf{S}I \to \mathsf{T}I$ but a family of natural transformations $R_f : R_J \mathsf{S}_f \dot{\to} \mathsf{T}_f R_I$. Subsequently, the dual of the above diagrams commute.

Proposition 27. Let $\mathbb{I}$ and $\mathbb{C}$ be categories and $\mathsf{S},\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ be $\mathbb{I}$-graded monads.

A *homomorphism* $\gamma : \mathsf{T} \to \mathsf{S}$ between the $\mathbb{I}$-graded monads is a left-lax natural transformation $L : \mathsf{S} \dot{\to} \mathsf{T}$ (since S and T map into a sub-category of **Cat** with $\mathsf{S}I = \mathsf{T}I = \mathbb{C}$) with $L_I = \mathsf{Id}_{\mathbb{C}}$. Therefore we have a family of natural transformations $\gamma_f = L_f : \mathsf{T}_f L_I \dot{\to} L_J \mathsf{S}_f$ which are natural transformations $\mathsf{T}_f \dot{\to} \mathsf{S}_f$ since L_I and L_J are the identity functor, with the following homomorphism axioms following from the definition of lax naturality (eliding again L_I and L_J which are identities):



Thus we can define the category of $\mathbb{I}$-graded monads over $\mathbb{C}$ base with these morphisms.

Lax natural transformations are also key to the next step of capturing *parameterised monads*.

6 Parameterised monads and generalised units

The notion of a monad with two indices which denote pre- and post-conditions was first proposed for the continuation monad by Wadler [1994]. Later, Atkey [2009b] generalised this idea, introducing the concept of a *parameterised monad* with a doubly-indexed functor $\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C},\mathbb{C}]$ which can, for example, model effects with Floyd-Hoare-logic reasoning via indices of pre- and post-conditions.

Definition 28. (Atkey [2009b])[5] A *parameterised monad* comprises a functor $\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C},\mathbb{C}]$ and natural transformations $\eta^{\mathsf{P}}_I : \mathsf{Id}_{\mathbb{C}} \dot{\to} \mathsf{P}(I,I)$ and $\mu^{\mathsf{P}}_{I,J,K} : \mathsf{P}(I,J)\mathsf{P}(J,K) \dot{\to} \mathsf{P}(I,K)$ satisfying analogous unit and associativity axioms to the usual monad axioms (with the addition of the indexing).

Furthermore, η^{P} is dinatural in I and $\mu^{\mathsf{P}}_{I,J,K}$ is dinatural in J and natural in I, K, equating to the following dinaturality diagrams for all $f : I \to J$ and $g : J \to J'$:

$$\begin{array}{ccc} \mathsf{P}(I,J)\,\mathsf{P}(J',K) & \xrightarrow{\mathsf{P}(I,g)\,\mathsf{P}(J',K)} & \mathsf{P}(I,J')\,\mathsf{P}(J',K) \\ {\scriptstyle \mathsf{P}(I,J)\mathsf{P}(g,K)}\downarrow & & \downarrow{\scriptstyle \mu^{\mathsf{P}}_{I,J',K}} \\ \mathsf{P}(I,J)\,\mathsf{P}(J,K) & \xrightarrow[\mu^{\mathsf{P}}_{I,J,K}]{} & \mathsf{P}(I,K) \end{array} \qquad \begin{array}{ccc} \mathsf{Id} & \xrightarrow{\eta^{\mathsf{P}}_I} & \mathsf{P}(I,I) \\ {\scriptstyle \eta^{\mathsf{P}}_J}\downarrow & & \downarrow{\scriptstyle \mathsf{P}(I,f)} \\ \mathsf{P}(J,J) & \xrightarrow[\mathsf{P}(f,J)]{} & \mathsf{P}(I,J) \end{array} \tag{2}$$

Multiplication μ^{P} requires that the post-condition of the outer computation matches the pre-condition of the inner computation. Thus, similarly to category-graded monads, parameterised monads restrict the composition of computations. Where category-graded monads differ is that this restriction is provided via morphisms, whereas the indices of parameterised monads are just pairs. Thus, category-graded monads can have indices with more computational content, e.g., proofs (see Example 40).

Example 29. [Atkey, 2009b, §2.3.2] Mutable state can be modelled by the state monad with $\mathsf{T}A = (A \times S)^S$ where S represents the type of the state. A parameterised monad provides a type-refined version of this where $\mathsf{P}\,(S_1,S_2)\,A = (A \times S_2)^{S_1}$ modelling the type of state at the start (S_1) and end of a computation (S_2). The parameterised monad operations have the same definition as the state monad operations $\eta^{\mathsf{P}}_S = x \mapsto \lambda s.(x,s)$ and $\mu^{\mathsf{P}}_{S_1,S_2,S_3} = c \mapsto \lambda s_1.\mathsf{app}\,(c\,s_1)$ where $c : (((A \times S_3)^{S_2}) \times S_2)^{S_1}$. Reading and writing from the store is provided by two families of operations: $\mathit{read}_S : \mathsf{P}(S,S)\,S$ and $\mathit{store}_{S_0,S} : S \to \mathsf{P}(S_0,S)\,1$.

This idea of using lax functors to generalise parameterised monads has been conjectured before by Capriotti [2015] in a blog post, but without further details. Here we show that, on there own, category-graded monads (lax functors) do not subsume parameterised monads, but they instead capture a subset of parameterised monads restricted to discrete indexing categories (with only identity morphisms). We show that discrete parameterised monads are category-graded monads (Section 6.1), and thus similar in power to monoid-graded monads (i.e., without ordering). We then add the additional structure of *generalised units* to category-graded monads (which arises as a kind of lax natural transformation) which accounts for the additional power of full parameterised monads (Section 6.2).

6.1 Discrete parameterised monads are category-graded monads

Standard notions of *discrete* and *indiscrete* categories are key here. We thus recall their definitions:

Definition 30. A category is *discrete* if its only morphisms are identities.

The functor $\Delta : \mathbf{Cat} \to \mathbf{Cat}$ discretises a category by discarding all but the identities

Definition 31. A category is *indiscrete* if there is exactly one morphism between every pair of objects.

The functor $\nabla : \mathbf{Cat} \to \mathbf{Cat}$ maps a category to its indiscrete form by replacing morphisms with pairs of the source and target objects ("dominoes"), i.e., $\nabla(\mathbb{C})(a,b) = \{(a,b)\}$. Identities are pairs of identical objects and composition of (a,b) with (b,c) yields (a,c).

A standard result is that the Δ and ∇ functors arise from a single adjoint triple (see Appendix, Prop. 43). The symbol Δ is often used for the diagonalisation functor, and ∇ co-diagonalisation. We reuse the notation as discretisation is akin to restricting a category to the diagonal of the adjacency matrix of its morphisms.

[5] Atkey actually presents parameterised monads with a ternary functor $\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \times \mathbb{C} \to \mathbb{C}$; we use an isomorphic binary functor here mapping to $[\mathbb{C},\mathbb{C}]$ to reduce clutter and as this is akin to presentations of graded monads on functors $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to [\mathbb{C},\mathbb{C}]$.

Definition 32. A *discrete parameterised monad* is a parameterised monad indexed by a discrete category, with functor $\mathsf{P} : \Delta(\mathbb{I})^{\mathsf{op}} \times \Delta(\mathbb{I}) \to [\mathbb{C},\mathbb{C}]$ which has a degenerate morphism mapping which is always the identity $\mathsf{P}\,(id,id) = id : \mathsf{P}\,(I,I)A \to \mathsf{P}\,(I,I)A$ since there are only identity morphisms in $\Delta(\mathbb{I})$. Dinaturality squares (eq. 2) trivially hold as they collapse to identities.

Proposition 33. Every discrete parameterised monad is a category-graded monad.

Proof. Let $(\mathsf{P},\mu^{\mathsf{P}},\eta^{\mathsf{P}})$ be a discrete parameterised monad on $\mathsf{P} : \Delta(\mathbb{I})^{\mathsf{op}} \times \Delta(\mathbb{I}) \to [\mathbb{C},\mathbb{C}]$. There is then a category-graded monad $\mathsf{T} : \nabla(\mathbb{I})^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$, graded by the indiscrete version of $\mathbb{I}$ which has pairs of objects as morphisms, and thus the morphism mapping of T is defined $\mathsf{T}\,(I,J) = \mathsf{P}\,(I,J)$ with operations:

$$\eta_I : \mathsf{Id} \dot{\to} \mathsf{T}_{(I,I)} = \eta^{\mathsf{P}}_I \qquad \mu_{(I,J),(J,K)} : \mathsf{T}_{(I,J)}\mathsf{T}_{(J,K)} \dot{\to} \mathsf{T}_{(I,K)} = \mu^{\mathsf{P}}_{I,J,K} \quad \textit{(for all } (I,J),(J,K) \in \nabla(\mathbb{I})_1\textit{)}$$

This construction is injective, with an inverse mapping from indiscrete-category-graded monads to discrete parameterised monads. Each unique morphism of an indiscrete category $\mathbb{J}$ can be identified by its singleton homset $\mathbb{J}(I,J)$. Thus, there is a map from indiscrete-category-graded monads $\mathsf{S} : \mathbb{J}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ to discrete parameterised monads $\mathsf{P} : \Delta(\mathbb{J})^{\mathsf{op}} \times \Delta(\mathbb{J}) \to [\mathbb{C},\mathbb{C}]$ where $\mathsf{P}\,(I,J)A = \mathsf{S}\,(\mathbb{J}(I,J))A$ and $\mathsf{P}\,(I,J)\,f = \mathsf{S}_{\mathbb{J}(I,J)}f$, and operations $\eta^{\mathsf{P}}_I = \eta_I$ and $\mu^{\mathsf{P}}_{I,J,K} = \mu_{\mathbb{J}(I,J),\mathbb{J}(J,K)}$. □

6.2 Parameterised monads are category-graded monads with a generalised unit

Of course, parameterised monads need not be discrete and therefore may have non-degenerate morphism mappings $\mathsf{P}\,(f,g)A : \mathsf{P}\,(I,J)A \to \mathsf{P}\,(I',J')A$ for $f : I' \to I$ and $g : J \to J'$ in $\mathbb{I}$. Through the Floyd-Hoare perspective, this morphism mapping corresponds to pre-condition strengthening (via f) and post-condition weakening (via g), i.e., a kind of approximation on the indices. Whilst object pairs $\mathbb{I}^{\mathsf{op}} \times \mathbb{I}$ do not contain computational content in the way that a morphism may (e.g., if it is a function), the morphism mapping provides a way to change an effectful computation via the morphisms of $\mathbb{I}$.

A category-graded monad could be constructed from a parameterised monad following the approach for discrete parameterised category-graded monads with $\mathsf{T} : \nabla(\mathbb{I})^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ defined on morphisms as $\mathsf{T}\,(I,J) = \mathsf{P}\,(I,J)$ (Prop 33). However, such a construction is non-injective since the morphism mapping of P is discarded. Alternatively, we might try to define $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ with $\mathsf{T}(f : I \to J) = P(I,J)$. The morphism mapping $\mathsf{P}(g,f) : \mathsf{P}(I,J) \dot{\to} \mathsf{P}(I',J')$ for $g : J \to J'$, $f : I' \to I$ would then correspond to a family of natural transformations of the form $\alpha_{f,g,k} : \mathsf{T}\,k \dot{\to} \mathsf{T}(g \circ k \circ f)$ with $k : I \to J$. However, no such family is elicited by a (2-)category-graded monad alone. Furthermore, $\mathsf{T}(f : I \to J) = \mathsf{P}(I,J)$ is non-injective since it maps to $\mathsf{P}(I,J)$ only when there is a morphism $I \to J \in \mathbb{I}_1$ despite the P object mapping being defined for all object pairs $\mathbb{I}^{\mathsf{op}} \times \mathbb{I}$.

Additional structure is therefore needed to capture full parameterised monads. For this, we introduce the notion of a *generalised unit*.

Definition 34. Let $(\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C}),\mu,\eta)$ be an $\mathbb{I}$-graded monad and let $\mathbb{S}$ be a *wide subcategory* $\mathbb{S} \subseteq \mathbb{I}$ where $\mathbb{S}_0 = \mathbb{I}_0$ (all the same objects) and $\iota : \mathbb{S} \to \mathbb{I}$ is the inclusion functor.

A *generalised unit* augments a category-graded monad with a right lax natural transformation (Def. 26) $R : \mathbf{Id} \dot{\to} (\mathsf{T} \circ \iota)$ mapping from the identity category-graded monad (Example 19). The right lax natural transformation has $R_I = \mathsf{Id}_{\mathbb{C}}$ for all I and thus its family of maps has signature $R_f : A \to \mathsf{T}_{\iota(f)}A$. Subsequently, the right lax natural transformation axioms are specialised to the following axioms, where we let

$\bar{\eta}_f = R_f$ and make the inclusion functor ι implicit:

$$\begin{array}{ccc} A & \xrightarrow{\bar{\eta}_f} & \mathsf{T}_f A \\ {\scriptstyle \bar{\eta}_{g\circ f}}\downarrow & & \downarrow{\scriptstyle \mathsf{T}_f \bar{\eta}_g} \\ \mathsf{T}_{g\circ f} A & \xleftarrow[\mu_{f,g}]{} & \mathsf{T}_f \mathsf{T}_g A \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{\eta_I} & \mathsf{T}_{id_I} A \\ & {\scriptstyle \bar{\eta}_{id_I,A}}\searrow & \downarrow{\scriptstyle id_{\mathsf{T}_{id_I A}}} \\ & & \mathsf{T}_{id_I} A \end{array}$$

The left square states that "generating" two computations indexed by morphisms in $\mathbb{S}_1$ via $\bar{\eta}$ and multiplying via μ is equivalent to generating via $\bar{\eta}$ the computation indexed by the composition of the indices. The right square is well defined by the requirement that $\mathbb{S}_0 = \mathbb{I}_0$, thus all identity morphisms of $\mathbb{I}$ are in $\mathbb{S}$, and $\bar{\eta}_{id_I}$ and η_I coincide.

We refer to the family of maps $\bar{\eta}_{f,A} : A \to \mathsf{T}_f A$ as the *generalised unit*, with the notation alluding to η, since it has a similar form to unit but is defined for all $f : I \to J \in \mathbb{S}_1$ rather than just on identities.

Example 35. Every category-graded monad has a generalised unit with $\mathbb{S} = \Delta(\mathbb{I})$, i.e., the subcategory containing only the identity morphisms. Since $\Delta(\mathbb{I})$ contains only the identity morphisms, $\bar{\eta}$ need only be defined on identities with $\bar{\eta}_{id_I} = \eta_I : \mathsf{Id}_{\mathbb{C}} \dot{\to} \mathsf{T}_{id_I}$.

Example 36. Consider a pomonoid-graded monad T presented as a 2-category-graded monad (Proposition 25) with $(M, \bullet, e, \sqsubseteq)$ and $\mathsf{T} : 1(M) \to \mathbf{Endo}(\mathbb{C})$. If the unit element is the bottom element of the ordering, i.e., $\forall m \in M . e \sqsubseteq m$, then there is a generalised unit with $\mathbb{S} = \mathbb{I}$ defined $\bar{\eta}_m = \mathsf{T}(e \sqsubseteq m) \circ \eta : \mathsf{Id}_{\mathbb{C}} \dot{\to} \mathsf{T}_m$.

Proposition 37. Every parameterised monad is a category-graded monad with a generalised unit.

Proof. Let $(\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C}, \mathbb{C}], \mu^{\mathsf{P}}, \eta^{\mathsf{P}})$ be a parameterised monad. From $\mathbb{I}$ we construct a category $\mathbb{I}^{\nabla}$ called the *pair completion* of $\mathbb{I}$ where objects are the objects of $\mathbb{I}$, and morphisms from I to J are either a morphism in $\mathbb{I}(I,J)$ or a pair (I,J), that is, we have homsets $\mathbb{I}^{\nabla}(I,J) = \mathbb{I}(I,J) \uplus \{(I,J)\}$. Thus, the homsets are the disjoint union of $\mathbb{I}$ morphisms or a pair (I,J), with injections into it written in_1 and in_2. Identities of $\mathbb{I}^{\nabla}$ are by $in_1(id_I)$ and composition is defined:

$$(g : J \to K) \circ (f : I \to J) = \begin{cases} in_1(g' \circ f') & f = in_1(f') \wedge g = in_1(g') \\ in_2(I,K) & \textit{otherwise} \end{cases} \tag{3}$$

We then define a category-graded monad $\mathsf{T} : (\mathbb{I}^{\nabla})^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ where $\mathsf{T}(f : I \to J) = \mathsf{P}\ (I,J)$ with operations: $\eta_I = \eta^{\mathsf{P}}_I$ and $\mu_{f,g} = \mu^{\mathsf{P}}_{I,J,K}$ for $f : I \to J, g : J \to K \in \mathbb{I}^{\nabla}{}_1$. The source and target objects I, J, and K are used to index μ^{P} without needing to determine whether the morphisms f and g are in the left or right injection of $\mathbb{I}^{\nabla}$.

Let $\mathbb{S} = \mathbb{I}$ which is a subcategory of $\mathbb{I}^{\nabla}$ via the inclusion which is the identity on objects and left injection in_1 on morphisms. This satisfies the property that all identities of $\mathbb{I}^{\nabla}$ are in $\mathbb{S}$ as identities are given by $in_1\, id$. Generalised unit $\bar{\eta}$ then has two equivalent definitions:

$$\forall f : I \to J \in \mathbb{I}_1 .\ \ \bar{\eta}_f = \mathsf{P}\, id_I\, f \circ \eta^P_I = \mathsf{P}\, f\, id_J \circ \eta^P_J \tag{4}$$

These two definitions of $\bar{\eta}$ are equivalent by dinaturality of η^{P} (the right equality *is* the dinaturality condition). The generalised unit axioms follow from bifunctor axiom $\mathsf{P}\, id_I\, id_I = id$ and dinaturality of μ^{P}.

In the Appendix, Proposition 44 shows that the above construction has a left inverse, i.e., *every category-graded monad with generalised unit indexed by* $\mathbb{I}^{\nabla}$ *with* $\mathbb{S} = \mathbb{I}$ *has a corresponding parameterised monad*. We give an outline here.

For a category-graded monad on $\mathsf{T} : (\mathbb{I}^{\nabla})^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ with $\bar{\eta}$ and subcategory $\mathbb{S} = \mathbb{I}$ we can construct a parameterised monad with functor $\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C},\mathbb{C}]$ on objects as $\mathsf{P}\,(I,I) = \mathsf{T}\,(in_1 id_I)$ and $\mathsf{P}\,(I,J) = \mathsf{T}\,(in_2\,(I,J))$ (for $I \neq J$). The morphism mapping $\mathsf{P}(f,g)\,h$ is built from μ and $\bar{\eta}$, where for all $f : I' \to I, g : J \to J' \in \mathbb{I}_1$, $h : A \to B \in \mathbb{C}_1$ (with shorthand $k = in_2(I,J)$, $\bar{f} = in_1\, f$ and $\bar{g} = in_1\, g$):

$$\mathsf{T}_k A \xrightarrow{\bar{\eta}_{\bar{f}}\mathsf{T}_k} \mathsf{T}_{\bar{f}}\mathsf{T}_k A \xrightarrow{\mathsf{T}_{\bar{f}}\mathsf{T}_k\bar{\eta}_{\bar{g}}} \mathsf{T}_{\bar{f}}\mathsf{T}_k\mathsf{T}_{\bar{g}} A \xrightarrow{\mu_{\bar{f},k,\mathsf{T}\bar{g}}} \mathsf{T}_{(k\circ\bar{f})}\mathsf{T}_{\bar{g}} A \xrightarrow{\mu_{k\circ\bar{f},\bar{g}}} \mathsf{T}_{(\bar{g}\circ k\circ\bar{f})} A \xrightarrow{\mathsf{T}h} \mathsf{T}_{(\bar{g}\circ k\circ\bar{f})} B$$

Bifunctoriality of P follows from the right lax natural transformation and category-graded-monads axioms. The parameterised monad operations are provided by category-graded monad $\mu^{\mathsf{P}}_{I,J,K} = \mu_{in_2(I,J),in_2(J,K)}$ and generalised unit $\eta^{\mathsf{P}}_I = \bar{\eta}_{in_1 id_I}$.

This construction is the inverse of the former, thus there is just one category-graded monad with generalised unit for every parameterised monad. □

Corollary 38. 2-cat-graded monads with a generalised unit subsume graded and parameterised monads.

The mapping of a parameterised monad to our structure here identifies two parts of a parameterised monad: the object mapping of P, defined for all pairs of objects and the morphism mapping of P, defined for all morphisms. These two classes are grouped into a single category via $\mathbb{I}^{\nabla}$ such that the generalised unit is defined only on the actual morphisms of $\mathbb{I}$, corresponding to the morphism mapping part of P.

Remark 39. Atkey [2009b] describes parameterised monads in a similar way to our description of category-graded monads: "*parameterised monads are to monads as categories are to monoids*". He illustrates this by generalising the writer monad $\mathsf{T}A = M \times A$ for some monoid on M, replacing M with morphisms of a small category. That is, for some small category S then $\mathsf{P} : S^{\mathsf{op}} \times S \to [\mathbf{Set},\mathbf{Set}]$ is defined $\mathsf{P}(I,J)A = S(I,J) \times A$, i.e., the set of $I \to J$ morphisms paired with the set A. Then $\eta^{\mathsf{P}}_{I,A} a \mapsto (id_I, a)$ and $\mu^{\mathsf{P}}_{I,J,K,A}(f,(g,a)) \mapsto (g \circ f, a)$. This construction is essentially a value-level version of what the indices of a category-graded monad provide. Thus category-graded monads provide a static trace of morphism composition (recall Example 20), whilst parameterised monads can give only a dynamic, value-level trace.

Example 40. *Making parameterised monads constructive via a category-graded monad.* We define a class of category-graded monads based on parameterised monads, but that are *not* parameterised monads. A parameterised monad $(\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C},\mathbb{C}], \mu^{\mathsf{P}}, \eta^{\mathsf{P}})$ induces a cat-graded monad on $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ with $\mathsf{T}\,(f : I \to J) = \mathsf{P}\,(I,J)$, i.e., source and target objects of f provide the parameterised monad indices with operations $\eta_I = \eta^{\mathsf{P}}_I$ and $\mu_{f,g} = \mu^{\mathsf{P}}_{I,J,K}$ (for $f : I \to J$, $g : J \to K$) and a generalised unit $\bar{\eta}_f = \mathsf{P}\, id_I\, f \circ \eta^{\mathsf{P}}_I$. This gives a restricted view of the parameterised monad P, allowing computations $\mathsf{P}\,(I,J)$ to be used only when there is a morphism (e.g., a proof or "path") $I \to J \in \mathbb{I}_1$. We thus call this a *constructive* parameterised monad. This example is only possible with the additional power of category-graded monads.

7 Example: combining graded monads and parameterised monads

Since 2-category-graded monads with generalised units unify both parameterised and graded monads (Corollary 38) then they can provide a model for systems which combine both quantitative reasoning and program logics into a single structure. For example, Barthe et al. [2016] define a probabilistic Hoare Logic (aHL) which provides a quantitative analysis of the "union bound" of probabilistic computations. Judgments in aHL for a program c are of the form: $\vdash_\beta c : \phi \Rightarrow \psi$ where the initial state of a program c satisfies pre-condition ϕ and after execution produces a distribution of states for which ψ holds. The

annotation β is the maximum probability that ψ does not hold. Derivations of judgments track the change in this probability bound β by structure on the annotation (where $\beta, \beta' \in [0,1]$ and $+$ is saturating at 1):

$$\frac{}{\vdash_0 \mathbf{skip} : \phi \Rightarrow \phi}(skip) \quad \frac{}{\vdash_0 x \leftarrow e : \phi[e/x] \Rightarrow \phi}(assgn) \quad \frac{\vdash_\beta c : \phi \Rightarrow \phi' \quad \vdash_{\beta'} c' : \phi' \Rightarrow \phi''}{\vdash_{\beta+\beta'} c;c' : \phi \Rightarrow \phi''}(seq)$$

$$\frac{\models \phi' \to \phi \quad \vdash_\beta c : \phi \Rightarrow \psi \quad \models \psi \to \psi' \quad \beta \leq \beta'}{\vdash_{\beta'} c : \phi' \Rightarrow \psi'}(weak)$$

The consequence rule (called *weak* above) combines pre-condition strengthening and post-condition weakening with approximation of the probability upper-bound.

We give a 2-category-graded monadic semantics to this system, using a product of two indexing categories: the 2-category of the additive monoid over $[0,1]$ (with ordering $\leq$) and the category $\mathbf{Prop}^\nabla$ whose morphisms are pairs of propositions **Prop** and logical implications. For brevity we consider just the subset of the aHL system given above, though the rest of the system can be readily modelled.

Barthe *et al.* interpret programs as functions from a store type State to distributions over stores $\mathbf{Distr}(\mathsf{State})$. We extend this to include a return value of type A denoting $\mathsf{D}_A = \mathsf{State} \to \mathbf{Distr}(\mathsf{State} \times A)$. We define a 2-category-graded monad on **Set** as a lax functor $\mathsf{T} : ([0,1] \times \mathbf{Prop}^\nabla)^{\mathsf{op}} \to [\mathbf{Set}, \mathbf{Set}]$ with generalised unit, combining Atkey's parameterised monad for a program logic (which refines the set of denotations to valid store transformations [Atkey, 2009b, p.27]) with the additional validity requirements with regards probabilities in aHL (the probability that the return state does not satisfy the post-condition is bounded above by β) where:

$$\mathsf{T}_{(\beta, f:\phi\to\psi)}A = \{c \in \mathsf{D}_A \mid \forall s_1.\ s_1 \models \phi \Rightarrow \exists (s_2, a).((s_2,a) \in c(s_1) \wedge s_2 \models \psi \wedge \Pr_{s_2}[\neg\psi] \leq \beta)\}$$

The definition of the category-graded monad operations is essentially that of a state monad combined with a distribution monad (e.g., $\eta_{(0,\phi),A} : A \to \mathsf{T}_{0,id_\phi:\phi\to\phi}A = x \mapsto \lambda s.\lambda p.(s,x)$) but whose set of values is refined by the validity requirements of the above definition. The generalised unit operation is defined when $\models \phi' \to \phi$ then $\bar{\eta}_{(0,f:\phi'\to\phi)} : A \to \mathsf{T}_{(0,f)}A = x \mapsto \lambda s.\lambda p.(s,x)$ where f is the model of the implication in **Prop**. Derivations are then interpreted as morphisms $[\![\vdash_\beta c : \phi \Rightarrow \psi]\!] : 1 \to \mathsf{T}_{\beta,f:\phi\to\psi}1$ with:

$$[\![\mathbf{skip}]\!] = \eta_{(0,\phi),1} : 1 \to \mathsf{T}_{0,id_\phi:\phi\Rightarrow\phi}1 \qquad [\![c;c']\!] = \mu_{(\beta,f),(\beta',g),1} \circ \mathsf{T}_{\beta,f:\phi\to\phi'}[\![c']\!] \circ [\![c]\!]$$

The (weak) rule is modelled by generalised unit and the approximation maps of the 2-cat-graded monad:

$$\mu_{(\beta',g\circ f),(0,g')} \circ \mathsf{T}_{g\circ f}\bar{\eta}_{(0,g':\phi'\to\phi)} \circ \mu_{(0,g),(\beta',f)} \circ \bar{\eta}_{(0,g:\psi\to\psi')} \circ \mathsf{T}_{\mathbf{f}:\beta\leq\beta'} \circ ([\![c]\!] : 1 \to \mathsf{T}_{\beta,f:\phi\to\psi}1)$$

8 Discussion

8.1 A more direct relationship between parameterised and graded monads

The main motivation for category-graded monads is to unify graded and parameterised monads. However, in some restricted cases, some parameterised monads can be mapped to graded monads more directly.

Proposition 41. For a parameterised monad $(\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C},\mathbb{C}], \mu^{\mathsf{P}}, \eta^{\mathsf{P}})$ where $\mathbb{I}$ is a monoidal category $(\mathbb{I}, \bullet, e)$ and $\mathbb{C}$ is *finitely complete* (in that it has all finite limits) there is a graded monad given by an *end* in $\mathbb{C}$ (generalising universal quantification) $\mathsf{G}f = \int_i \mathsf{P}(i, i \bullet f)$. with unit $\eta : A \to \mathsf{G}e = \eta^{\mathsf{P}} : A \to \mathsf{P}(i, i \bullet e)$ and graded monad multiplication as follows (which shows some calculation):

$$\mathsf{G}f\,\mathsf{G}g = \int_i \mathsf{P}(i, i\bullet f) \int_j \mathsf{P}(j, j \bullet g) = \int_i \mathsf{P}(i, i\bullet f)\mathsf{P}(i\bullet f, (i\bullet f)\bullet g) \xrightarrow{\mu^{\mathsf{P}}} \int_i \mathsf{P}(i, (i\bullet f)\bullet g) = \int_i \mathsf{P}(i, i\bullet(f\bullet g)) = \mathsf{G}(f\bullet g)$$

Mapping the other way (graded into parameterised) is more difficult as two indices are needed from one.

8.2 Categorical semantics

Given a notion of tensorial strength for category-graded monads, it is straightforward to define a calculus whose denotational model is given by the category-graded monad operations, i.e., a language like the monadic metalanguage of Moggi [1991] (akin to Haskell's *do*-notation) for sequential composition of effectful computations. The core sequential composition and identity rules are then of the form:

$$\frac{\Gamma \vdash t : \mathsf{T}_f A \quad \Gamma, x : A \vdash t' : \mathsf{T}_g B \quad f : I \to J \quad g : J \to K}{\Gamma \vdash \mathbf{let}\ x \leftarrow t\ \mathbf{in}\ t' : \mathsf{T}_{g \circ f} B} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash \langle t \rangle : \mathsf{T}_{id_I} A}$$

The semantics of such a calculus resembles that of the monadic meta language, taking the same form but with the added morphism grades. This semantics then requires a notion of tensorial strength in the category-graded setting, that is a natural transformation $\sigma_{f,A,B} : A \times \mathsf{T}_f B \to \mathsf{T}_f (A \times B)$ for all $f : I \to J \in \mathbb{I}_1$ satisfying a graded variant of the usual monadic strength axioms. Tensorial strength has been previously considered for graded [Katsumata, 2014] and parameterised monads [Atkey, 2009b]. Defining a subsuming notion of *strong category-graded monads* as above appears to be straightforward.

8.3 Further work

Category-graded comonads Various works employ *graded comonads* to give the semantics of *coeffects* (Petricek et al. [2013], Brunel et al. [2014], Ghica and Smith [2014]). Category-graded monads dualise straightforwardly to *category-graded comonads* as a *colax* functor $\mathsf{D} : \mathbb{I} \to \mathbf{Endo}(\mathbb{C})$ witnessed by natural transformations $\varepsilon_I : \mathsf{D}_{id_I} \dot{\to} \mathsf{Id}$ and $\delta_{f,g} : \mathsf{D}_{g \circ f} \dot{\to} \mathsf{D}_g \mathsf{D}_f$ with dual axioms to category-graded monads. The source of the colax functor is dual to the $\mathbb{I}^{\mathsf{op}}$ source of a category-graded monad lax functor.

Graded comonads are category-graded comonads however graded comonads usually have a semiring structure on their indices, adding extra graded monoidal structure (called *exponential graded comonads* [Gaboardi et al., 2016]). Further work is to incorporate such additional structure into our formulation.

Polymonads, supermonads, and productoids *Polymonads* provide a programming-oriented generalisation of monads replacing the single type constructor M of a monad with a family of constructors Σ (Hicks et al. [2014]). For some triples of $\mathsf{M}, \mathsf{N}, \mathsf{P} \in \Sigma$ there is a *bind* operation of type: $\forall a, b. \mathsf{M}\,\mathsf{a} \to (\mathsf{a} \to \mathsf{N}\,\mathsf{b}) \to \mathsf{P}\,\mathsf{b}$ which allows two effectful computations captured by M and N to be composed and encoded by P. This generalises the familiar *bind* operation for monadic programming of type $\forall a, b. \mathsf{M}\,\mathsf{a} \to (\mathsf{a} \to \mathsf{M}\,\mathsf{b}) \to \mathsf{M}\,\mathsf{b}$ which internalises Kleisli extension of a strong monad.

Bracker and Nilsson [2016] provide a similar structure to polymonads, called *supermonads* where the constructors $\mathsf{M}, \mathsf{N}, \mathsf{P}$ are derived from an indexed family. The additional power of supermonads is that each functor need not be an endofunctor, but instead maps from some subcategory of the base category. This provides models which have some type-predicate-like restrictions on functors.

Another related structure is the *productoid* of Tate [2013] with an indexed family of constructors $\mathsf{T} : \mathit{Eff} \to [\mathbb{C}, \mathbb{C}]$, indexed by an *effectoid* structure Eff which is a kind of relational ordered monoid. Tate mentions the possibility of modelling this 2-categorically.

Future work is to study a generalisation of these seemingly related structures.

Hoare and Dijkstra Monads Nanevski et al. [2008] introduced the notion of *Hoare monads* which resemble parameterised monads but indexed by pre-conditions that are dependent on a heap value and post-conditions which are dependent on a heap and the return value of a computation. This is generalised by the notion of *Dijkstra* monads [Swamy et al., 2013, Maillard et al., 2019]) indexed by a predicate transformer which, given a post-condition on the final heap and value, computes the weakest pre-condition

of the computation. Studying how these structures fit with other kinds of indexed generalisations of monads, perhaps through the lense of lax functors, is interesting further work.

Monads on indexed sets An alternate indexed generalisation of monads is to consider monads over indexed sets, as in the work of McBride [2011]. This provides a fine-grained model of effects which can react to external uncertainty in the program state in a more natural way than the approaches discussed here. Exploring the connection between monads on indexed sets/types and cat-graded monads is further work.

The alternate oidification Category-graded monads were presented as the oidification of monads as a lax functor $\mathsf{T} : \mathbf{1} \to \mathbf{Endo}(\mathbb{C})$ to $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$. An alternate, orthogonal oidification is to generalise the target category as well to the lax functor $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Cat}$ where each $\mathsf{T}(f : I \to J)$ is then a functor from category $\mathsf{T}I$ to $\mathsf{T}J$, rather than an endofunctor on a particular category.

Such a generalisation suggests further control over the models of effects where for a morphism $f : I \to J$ the categories $\mathsf{T}I$ and $\mathsf{T}J$ may differ. For example, they may be subcategories of some overall base category as in the supermonads of Bracker and Nilsson [2016]. Exploring this is future work.

Adjunctions Fujii et al. [2016] show how a graded monad can be factored into an adjunction with graded analogues of the Kleisli and Eilenberg-Moore constructions. Relatedly, Atkey [2009a] gives notions of Kleisli and Eilenberg-Moore category for parameterised monads. Defining the appropriate adjunctions which gives rise to category-graded monads is further work. This would provide the opportunity to consider different kinds of semantics, for example, for call-by-push-value (Levy [2006]).

8.4 Summary and concluding remarks

Following Bénabou's perspective of monads as degenerate lax functors $\mathsf{T} : \mathbf{1} \to \mathbf{Endo}(\mathbb{C})$ involving single object categories, we applied the notion of oidification, generalising the source category from a point to a category $\mathsf{T} : \mathbb{I}^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$. This yielded the base notion of category-graded monads from which we encompass the full power of graded and parameterised monads as found in the recent literature. Namely:

- Monoid-graded monads are category-graded monads (Prop. 23);
- Pomonoid-graded monads are 2-category graded monads (Prop. 25);
- Discrete parameterised monads are category-graded monads (Prop. 33);
- Parameterised monads are category-graded monads with a generalised unit (Prop. 37) $\bar{\eta}_f : \mathsf{Id} \dot{\to} \mathsf{T}_f$ (a right lax natural transformation) for all $f \in \mathbb{S}_1$ where $\mathbb{S} \subseteq \mathbb{I}$ and $\mathbb{S}_0 = \mathbb{I}_0$.

Each result is an injection into the more general structure. This could be made stronger by a full and faithful embedding, but this requires a definition of categories of parameterised monads and graded monads, but the former is not provided in the literature. This is future work.

Category-graded monads have the feel of a *proof relevant* version of a graded monad or parameterised monad where the index is not merely a value, but a computation, that is a program or a proof. We leave the door open for future applications to utilise this generality, not just to provide a framework for including both graded and parameterised monads into one cohesive whole, but also for programs and models with more fine-grained computational indices.

Acknowledgments Thanks to Bob Atkey, Iavor Diatchki, Shin-ya Katsumata, and Ben Moon for useful discussions and comments on a draft of this paper. Specific thanks to Iavor for pointing out the connection to the Grothendieck construction and Shin-ya for various mathematical insights. We also thank the reviewers for their helpful feedback.

References

R. Atkey. Algebras for parameterised monads. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 3–17. Springer, 2009a. doi:10.1007/978-3-642-03741-2_2.

R. Atkey. Parameterised notions of computation. *JFP*, 19(3-4):335–376, 2009b. doi:10.1017/S095679680900728X.

G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub. A program logic for union bounds. In *ICALP 2016*, pages 107:1–107:15, 2016. doi:10.4230/LIPIcs.ICALP.2016.107.

J. Bénabou. Introduction to bicategories. In *Reports of the Midwest Category Seminar*, pages 1–77. Springer, 1967. doi:10.1007/BFb0074299.

P. Bertozzini, R. Conti, and W. Lewkeeratiyutkul. A Horizontal Categorification of Gelfand Duality. *arXiv preprint arXiv:0812.3601*, 2008a. doi:10.1016/j.aim.2010.06.025.

P. Bertozzini, R. Conti, and W. Lewkeeratiyutkul. Non-Commutative Geometry, Categories and Quantum Physics. *East West J. Math.*, 2007(arXiv: 0801.2826):S213–S259, 2008b.

J. Bracker and H. Nilsson. Supermonads: one notion to bind them all. In *Proceedings of the 9th International Symposium on Haskell*, pages 158–169. ACM, 2016. doi:10.1145/2976002.2976012.

A. Brunel, M. Gaboardi, D. Mazza, and S. Zdancewic. A Core Quantitative Coeffect Calculus. In *ESOP*, pages 351–370, 2014. doi:10.1007/978-3-642-54833-8_19.

P. Capriotti. Monads as lax functors. `https://www.paolocapriotti.com/blog/2015/06/22/monads-as-lax-functors/index.html` (accessed March 2020), 2015.

S. Fujii, S. Katsumata, and P.-A. Melliès. Towards a Formal Theory of Graded Monads. In *FOSSACS*, pages 513–530. Springer, 2016. doi:10.1007/978-3-662-49630-5_30.

M. Gaboardi, S. Katsumata, D. Orchard, F. Breuvart, and T. Uustalu. Combining Effects and Coeffects via Grading. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 476–489, 2016. doi:10.1145/2951913.2951939.

D. R. Ghica and A. I. Smith. Bounded Linear Types in a Resource Semiring. In *European Symposium on Programming Languages and Systems*, pages 331–350. Springer, 2014. doi:10.1007/978-3-642-54833-8_18.

J. Gibbons. Comprehending Ringads. In *A List of Successes That Can Change the World*, pages 132–151. Springer, 2016. doi:10.1007/978-3-319-30936-1_7.

A. Grothendieck. *Catégories fibrées et descente*. Institut des Hautes Etudes Scientifiques, 1961.

M. Hicks, G. Bierman, N. Guts, D. Leijen, and N. Swamy. Polymonadic programming. *Electronic Proceedings in Theoretical Computer Science*, 153:7999, Jun 2014. ISSN 2075-2180. doi:10.4204/eptcs.153.7.

K. Imai, S. Yuen, and K. Agusa. Session Type Inference in Haskell. In *Proc. of PLACES*, pages 74–91, 2010. doi:10.4204/EPTCS.69.6.

B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1999. ISBN 978-0-444-50170-7.

S. Katsumata. Parametric effect monads and semantics of effect systems. In *Proceedings of POPL 2014*, pages 633–645. ACM, 2014. doi:10.1145/2535838.2535846.

P. B. Levy. Call-by-push-value: Decomposing call-by-value and call-by-name. *Higher-Order and Symbolic Computation*, 19(4):377–414, 2006. doi:10.1007/s10990-006-0480-6.

K. Maillard, D. Ahman, R. Atkey, G. Martínez, C. Hriţcu, E. Rivas, and É. Tanter. Dijkstra monads for all. *Proceedings of the ACM on Programming Languages*, 3(ICFP):104, 2019. doi:10.1145/3341708.

C. McBride. Functional pearl: Kleisli arrows of outrageous fortune. 2011.

P.-A. Mellies. Parametric monads and enriched adjunctions. Available via http://www.pps.univ-paris-diderot.fr/~mellies/tensorial-logic.html, 2012.

S. Milius, D. Pattinson, and L. Schröder. Generic Trace Semantics and Graded Monads. In L. S. Moss and P. Sobocinski, editors, *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*, volume 35 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 253–269, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-84-2. doi:10.4230/LIPIcs.CALCO.2015.253.

E. Moggi. Computational lambda-calculus and monads. In *Logic in Computer Science, 1989. LICS'89, Proceedings., Fourth Annual Symposium on*, pages 14–23. IEEE, 1989. doi:10.1109/LICS.1989.39155.

E. Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991. doi:10.1016/0890-5401(91)90052-4.

A. Mycroft, D. Orchard, and T. Petricek. Effect Systems Revisited – Control-Flow Algebra and Semantics. In *Semantics, Logics, and Calculi*, pages 1–32. Springer, 2016. doi:10.1007/978-3-319-27810-0_1.

A. Nanevski, G. Morrisett, and L. Birkedal. Hoare type theory, polymorphism and separation. *Journal of Functional Programming*, 18(5-6):865–911, 2008. doi:10.1017/S0956796808006953.

nLab authors. Horizontal Categorification. http://ncatlab.org/nlab/show/horizontalcategorification, Jan. 2020. Revision 20.

D. Orchard and N. Yoshida. Effects as Sessions, Sessions as Effects. *POPL*, 51(1):568–581, 2016. doi:10.1145/2914770.2837634.

D. Orchard and N. Yoshida. Session types with linearity in Haskell. In S. Gay and A. Ravara, editors, *Behavioural Types: from Theory to Tools*, pages 219–241. River Publishers, 2017. ISBN 9788793519824. doi:10.13052/rp-9788793519817.

D. Orchard, T. Petricek, and A. Mycroft. The semantic marriage of monads and effects. *CoRR*, abs/1401.5391, 2014. URL http://arxiv.org/abs/1401.5391.

D. Orchard, V.-B. Liepelt, and H. Eades III. Quantitative Program Reasoning with Graded Modal Types. *Proc. ACM Program. Lang.*, 3(ICFP), July 2019. doi:10.1145/3341714.

T. Petricek, D. Orchard, and A. Mycroft. Coeffects: Unified static analysis of context-dependence. In *ICALP (2)*, pages 385–397, 2013. doi:10.1007/978-3-642-39212-2_35.

R. Pucella and J. A. Tov. Haskell Session Types with (Almost) no Class. In *Proc. of Haskell symposium '08*, pages 25–36. ACM, 2008. doi:10.1145/1411286.1411290.

A. L. Smirnov. Graded monads and rings of polynomials. *Journal of Mathematical Sciences*, 151(3): 3032–3051, 2008. doi:10.1007/s10958-008-9013-7.

D. Stefan, A. Russo, J. C. Mitchell, and D. Mazières. *Flexible dynamic information flow control in Haskell*, volume 46. ACM, 2011. doi:10.1145/2034675.2034688.

R. Street. Two constructions on lax functors. *Cahiers de topologie et géométrie différentielle catégoriques*, 13(3):217–264, 1972.

N. Swamy, J. Weinberger, C. Schlesinger, J. Chen, and B. Livshits. Verifying higher-order programs with the Dijkstra monad. *ACM SIGPLAN Notices*, 48(6):387–398, 2013. doi:10.1145/2499370.2491978.

R. Tate. The sequential semantics of producer effect systems. In *ACM SIGPLAN Notices*, volume 48, pages 15–26. ACM, 2013. doi:10.1145/2429069.2429074.

P. Wadler. Monads and composable continuations. *Lisp and Symbolic Computation*, 7(1):39–55, 1994. doi:10.1007/BF01019944.

P. Wadler and P. Thiemann. The marriage of effects and monads. *ACM Trans. Comput. Logic*, 4:1–32, January 2003. doi:10.1145/601775.601776.

A Additional background and details

Definition 42. *2-categories* extend the notion of a category with morphism between morphisms. A 2-category $\mathbb{C}$ has a class of objects $\mathbb{C}_0$, a class of 1-morphisms (usual morphisms, between objects) $\mathbb{C}_1$, and a class of 2-morphisms (between 1-morphisms) $\mathbb{C}_2$. We write 2-morphisms in bold, e.g. $\mathbf{k} : f \Rightarrow g$ is a 2-morphism between 1-morphisms $f, g : A \to B$.

1-morphisms compose as usual via $\circ$ and have identities id_A for all objects A. 2-morphisms have two notions of composition: *horizontal*, written $\circ_0$, which composes along objects and *vertical* written $\circ_1$ which composes along morphisms. That is, horizontal composition of a 2-morphism $\mathbf{i} : f \Rightarrow g$ where $f, g : A \to B$ and $\mathbf{j} : f' \Rightarrow g'$ where $f', g' : B \to C$ yields $\mathbf{j} \circ_0 \mathbf{i} : (f' \circ f) \Rightarrow (g' \circ g)$. For morphisms $f, g, h : A \to B$ and 2-morphisms $\mathbf{i} : f \Rightarrow g$ and $\mathbf{j} : g \Rightarrow h$ then their vertical composition is $\mathbf{j} \circ_1 \mathbf{i} : f \Rightarrow h$.

Both vertical and horizontal composition are associative and have an identity via the identity 2-morphism $\mathbf{id}_f : f \Rightarrow f$. Additionally, vertical and horizontal composition satisfy the *interchange* axiom: $(\mathbf{i} \circ_1 \mathbf{j}) \circ_0 (\mathbf{k} \circ_1 \mathbf{l}) = (\mathbf{i} \circ_0 \mathbf{k}) \circ_1 (\mathbf{j} \circ_0 \mathbf{l})$.

Proposition 43. The following adjoint triple gives rise to functors for mapping (small) categories to their discrete and indiscrete versions:

$$\mathsf{d} \dashv \mathsf{ob} \dashv \mathsf{i} : \mathbf{Set} \to \mathbf{Cat}$$

where $\mathsf{d} : \mathbf{Set} \to \mathbf{Cat}$ maps a set to a discrete category (the set gives the objects of the category), $\mathsf{ob} : \mathbf{Cat} \to \mathbf{Set}$ maps a category to its set of objects, $\mathsf{i} : \mathbf{Set} \to \mathbf{Cat}$ maps a set to an indiscrete category with morphisms given by the unique pair of the objects, i.e., $(\mathsf{i}A)(a,b) = \{(a,b)\}$. Identities are pairs of identical objects and composition of (a,b) with (b,c) yields (a,c); think dominoes.

From this adjoint triple there arises two functors $\Delta = \mathsf{d} \circ \mathsf{ob} : \mathbf{Cat} \to \mathbf{Cat}$ which discretises categories by discarding all but the identities and $\nabla = \mathsf{i} \circ \mathsf{ob} : \mathbf{Cat} \to \mathbf{Cat}$ which maps a category to its indiscrete form by replacing their morphisms with pairs of objects.

The following provides the details of the inverse direction of Proposition 37.

Proposition 44. Every category-graded monad with generalised unit indexed by $\mathbb{I}^{\nabla}$ with $\mathbb{S} = \mathbb{I}$ has a corresponding parameterised monad.

Proof. Let $(\mathsf{T}, \eta, \mu, \bar{\eta})$ be a category-graded monad with generalised unit $\bar{\eta}$ and $\mathsf{T} : (\mathbb{I}^{\nabla})^{\mathsf{op}} \to \mathbf{Endo}(\mathbb{C})$ and subcategory $\mathbb{S} = \mathbb{I}$. Then, there is a parameterised monad with $\mathsf{P} : \mathbb{I}^{\mathsf{op}} \times \mathbb{I} \to [\mathbb{C}, \mathbb{C}]$ defined on objects

as $\mathsf{P}(I,I) = \mathsf{T}\ (in_1 id_I)$ and $\mathsf{P}(I,J) = \mathsf{T}\ (in_2\ (I,J))$ (for $I \neq J$) and on morphisms $f : I' \to I, g : J \to J' \in \mathbb{I}_1$, $h : A \to B \in \mathbb{C}_1$ (with shorthand $k = in_2(I,J)$, $\bar{f} = in_1\, f$ and $\bar{g} = in_1\, g$):

$$\mathsf{P}(f,g)\,h = \mathsf{T}_{\bar{g}\circ k\circ \bar{f}} h \circ \mu_{k\circ\bar{f},\bar{g}} \circ \mu_{\bar{f},k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_k \ \ : \mathsf{P}\,(I,J)A \to \mathsf{P}\,(I',J')\,B \tag{5}$$

The following shows part of this definition diagrammatically for clarity:

$$\mathsf{T}_k A \xrightarrow{\bar{\eta}_{\bar{f}}\mathsf{T}_k} \mathsf{T}_{\bar{f}}\mathsf{T}_k A \xrightarrow{\mathsf{T}_{\bar{f}}\mathsf{T}_k\bar{\eta}_{\bar{g}}} \mathsf{T}_{\bar{f}}\mathsf{T}_k\mathsf{T}_{\bar{g}}A \xrightarrow{\mu_{\bar{f},k,\mathsf{T}\bar{g}}} \mathsf{T}_{(k\circ\bar{f})}\mathsf{T}_{\bar{g}}A \xrightarrow{\mu_{k\circ\bar{f},\bar{g}}} \mathsf{T}_{\bar{g}\circ k\circ\bar{f}}A \xrightarrow{\mathsf{T}h} \mathsf{T}_{\bar{g}\circ k\circ\bar{f}}B$$

The morphism mapping of P is thus built from μ and $\bar{\eta}$ and is well defined since the functor $\mathsf{T}(\bar{g}\circ(in_2(I,J))\circ\bar{f})$ equals $\mathsf{T}(in_2(I',J'))$ by composition in $\mathbb{I}^{\nabla}$ (eq. 3). Thus, the domain of the functor matches the object mapping of $\mathsf{P}(I',J')$. Since $\mathbb{S} = \mathbb{I}$, $\bar{\eta}$ is defined for all morphisms of $\mathbb{I}$ which are wrapped by in_1 to indicate they are morphisms on $\mathbb{I}$ rather than object pairs.

Bifunctoriality of $\mathsf{P}(f,g)h$ holds as follows:

- $\mathsf{P}(f'\circ f, g'\circ g)(h'\circ h) = \mathsf{P}(f,g')h' \circ \mathsf{P}(f',g)h \ : \ \mathsf{P}(K',I)A \to \mathsf{P}(I',K)C$
  for all $h : A \to B, h' : B \to C \in \mathbb{C}_1$ and $g : I \to J, g' : J \to K \in \mathbb{I}_1$ and $f : I' \to J'$, $f' : J' \to K' \in \mathbb{I}_1$
  follows from the right lax natural transformation axiom on μ and associativity of μ, via the following reasoning (where $k = in_2(K,I')$ and $k' = in_2(J',J) = \bar{g}\circ k\circ\bar{f'}$ by the definition of composition for $\mathbb{I}^{\nabla}$ (eq.3), and a line above a morphism means that in_1 is applied to it):

$$\begin{aligned}
&\mathsf{P}(f,g')h' \circ \mathsf{P}(f',g)h \\
&\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}} h' \circ \mu_{k'\circ\bar{f},\bar{g}'} \circ \mu_{\bar{f},k'}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{k'}\,\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mathsf{T}_{\bar{g}\circ k\circ\bar{f}'} h \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\bar{\eta}\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}} h' \circ \mu_{k'\circ\bar{f},\bar{g}'} \circ \mu_{\bar{f},k'}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{k'}\,\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'} h \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\bar{\eta}\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}} h' \circ \mu_{k'\circ\bar{f},\bar{g}'} \circ \mu_{\bar{f},k'}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{k'}\mathsf{T}_{\bar{g}'} h \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}} h' \circ \mu_{k'\circ\bar{f},\bar{g}'} \circ \mathsf{T}_{k'\circ\bar{f}}\mathsf{T}_{\bar{g}'} h \circ \mu_{\bar{f},k'}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}} h' \circ \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}} h \circ \mu_{k'\circ\bar{f},\bar{g}'} \circ \mu_{\bar{f},k'}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{func.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}}(h'\circ h) \circ \mu_{k'\circ\bar{f},\bar{g}'} \circ \mu_{\bar{f},k'}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ assoc.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ k'} \circ \mathsf{T}_{\bar{f}}\mu_{k',\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{k\circ\bar{f}',\bar{g}} \circ \mu_{\bar{f}',k}\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ assoc.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ k'} \circ \mathsf{T}_{\bar{f}}\mu_{k',\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{k'} \circ \mu_{\bar{f}',\bar{g}\circ k} \circ \mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\bar{\eta}\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ k'} \circ \mathsf{T}_{\bar{f}}\mu_{k',\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{g}\circ k\circ\bar{f}'}\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{f}',\bar{g}\circ k} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{f'}\mathsf{T}_k\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ k'} \circ \mathsf{T}_{\bar{f}}\mu_{k',\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{f}',\bar{g}\circ k}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_{g\circ k}\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{f'}\mathsf{T}_k\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ k'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ k'} \circ \mathsf{T}_{\bar{f}}\mu_{k',\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{f}',\bar{g}\circ k}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_{g\circ k}\mathsf{T}_{\bar{g}'}\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{f'}\mathsf{T}_k\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{k'\ def.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{g}\circ k\circ\bar{f}',\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{f}',\bar{g}\circ k}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ assoc.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{f}',\bar{g}'\circ\bar{g}\circ k} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}\circ k}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ assoc.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f},\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'} \circ \mathsf{T}_{\bar{f}}\mu_{\bar{f}',\bar{g}'\circ\bar{g}\circ k} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mu_{k,\bar{g}\circ k}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mu_{\bar{g},\bar{g}'} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ assoc.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}\circ k} \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_{\bar{g}'\circ\bar{g}\circ k} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mu_{\bar{g},\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ assoc.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}\circ k} \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_{\bar{g}'\circ\bar{g}\circ k}\mathsf{T}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mu_{\bar{g},\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\bar{\eta}\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mu_{\bar{g},\bar{g}'} \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}'} \circ \mathsf{T}_{\bar{f}'}\mathsf{T}_k\,\bar{\eta}_{\bar{g}} \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{func.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k(\mu_{\bar{g},\bar{g}'} \circ \mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{g}}) \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\bar{\eta}\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_k\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k(\mu_{\bar{g},\bar{g}'} \circ \mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{g}}) \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\mu\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'\circ\bar{f}}\mathsf{T}_k(\mu_{\bar{g},\bar{g}'} \circ \mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{g}}) \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_k \circ \bar{\eta}_{\bar{f}}\mathsf{T}_{\bar{f}'}\mathsf{T}_k \circ \bar{\eta}_{\bar{f}'}\mathsf{T}_k \\
\{\bar{\eta}\ nat.\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'\circ\bar{f}}\mathsf{T}_k(\mu_{\bar{g},\bar{g}'} \circ \mathsf{T}_{\bar{g}}\bar{\eta}_{\bar{g}'} \circ \bar{\eta}_{\bar{g}}) \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_k \circ \mathsf{T}_f\bar{\eta}_{\bar{f}'}\mathsf{T}_k \circ \bar{\eta}_{\bar{f}}\mathsf{T}_k \\
\{\bar{\eta}/mu\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'\circ\bar{f}}\mathsf{T}_k\,\bar{\eta}_{\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f},\bar{f}'}\mathsf{T}_k \circ \mathsf{T}_f\bar{\eta}_{\bar{f}'}\mathsf{T}_k \circ \bar{\eta}_{\bar{f}}\mathsf{T}_k \\
\{\bar{\eta}/mu\} &\equiv \mathsf{T}_{\bar{g}'\circ\bar{g}\circ k\circ\bar{f}'\circ\bar{f}}(h'\circ h) \circ \mu_{k\circ\bar{f}'\circ\bar{f},\bar{g}'\circ\bar{g}} \circ \mu_{\bar{f}'\circ\bar{f},k}\mathsf{T}_{\bar{g}'\circ\bar{g}} \circ \mathsf{T}_{\bar{f}'\circ\bar{f}}\mathsf{T}_k\,\bar{\eta}_{\bar{g}'\circ\bar{g}} \circ \bar{\eta}_{\bar{f}'\circ\bar{f}}\mathsf{T}_k \\
&\equiv \mathsf{P}(f'\circ f, g'\circ g)(h'\circ h)
\end{aligned}$$

- $\mathsf{P}(id_I, id_I)id_A = id_{\mathsf{P}(I,I)A}$ follows from the right lax natural transformation axiom on η and the unit properties of cat-graded monads, via the following reasoning (where $k = in_2(I,I)$):

$$\begin{aligned}
& \mathsf{P}(id_I, id_I)id_A \\
\equiv\ & \mathsf{T}_{\bar{id}\circ k\circ \bar{id}}id \circ \mu_{k\circ\bar{id},\bar{id}} \circ \mu_{\bar{id},k}\mathsf{T}_{\bar{id}} \circ \mathsf{T}_{\bar{id}}\mathsf{T}_k\,\bar{\eta}_{\bar{id}} \circ \bar{\eta}_{\bar{id}}\mathsf{T}_k \\
\{\bar{\eta}/id\ property\} \equiv\ & \mathsf{T}_k id \circ \mu_{k,\bar{id}} \circ \mu_{\bar{id},k}\mathsf{T}_{\bar{id}} \circ \mathsf{T}_{\bar{id}}\mathsf{T}_k\,\eta_I \circ \eta_I\mathsf{T}_k \\
\{\mu\ naturality\} \equiv\ & \mathsf{T}_k id \circ \mu_{k,\bar{id}} \circ \eta_I \circ \mu_{\bar{id},k}\mathsf{T}_{\bar{id}} \circ \eta_I\mathsf{T}_k \\
\{\mu\ unitality\} \equiv\ & \mathsf{T}_k id \circ \mu_{k,\bar{id}} \circ \eta_I\mathsf{T}_k \\
\{\mu\ unitality\} \equiv\ & \mathsf{T}_k id \\
\{functor\ identity\ property\} \equiv\ & id_{\mathsf{T}_k} \\
\equiv\ & id_{\mathsf{P}(I,I)A}
\end{aligned}$$

The parameterised monad operations follow from the cat-graded monad $\mu^{\mathsf{P}}_{I,J,K} = \mu_{in_2(I,J),in_2(J,K)}$ and generalised unit $\eta^{\mathsf{P}}_I = \bar{\eta}_{in_1 id_I}$. This mapping is inverse to the former, e.g., with $\eta^{\mathsf{P}}_I = \bar{\eta}_{in_1(id_I)} = \mathsf{P}id_I id_I \circ \eta^{\mathsf{P}}_I = \eta^{\mathsf{P}}_I$ (μ is more direct). For $\mathsf{P}\,(f,g)\,h$, substituting eq. 4 into eq. 5 and applying the category-graded monad laws yields $\mathsf{P}(f,g)h$ (calculation elided for brevity). Therefore, there is just one category-graded monad with generalised unit for every parameterised monad. □