Unifying graded and parameterised monads

Dominic Orchard

Philip Wadler

Harley Eades III

MSFP 2020
The humble monad

\[ T : \mathbb{C} \rightarrow \mathbb{C} \]

\[ T \circ T \quad \text{Id} \]

Join :: m m a -> m a

\[ \mu \]  

\[ \eta \]

Return :: a -> m a

+ associativity and unitality axioms

multiplication  

unit
Graded monads

\[ G : \mathcal{E} \rightarrow [\mathbb{C}, \mathbb{C}] \]  

 Functor

 (Discrete) monoidal category

\[ G(x \cdot y) \]

 multiplication

\[ \mu_{x,y} \]

 associativity and unitality axioms

\[ G I \]

 unit

\[ \eta \]

\[ \text{Id} \]

\[ \mu_{x,y} \]

[Katsumata'14 - Parametric effect monads and semantics of effect systems]

[Wadler&Thiemman'03 - Marriage of effects and monads]
Graded monads for type-based effect analysis

monadic metalanguage

\[
\Gamma \vdash e : MA \quad \Gamma, x : A \vdash e' : MB \\
\Gamma \vdash \text{do } x \leftarrow e; e' : MB \\
\Gamma \vdash \text{return } e : MA
\]
Graded monads for type-based effect analysis

\[
\Gamma \vdash e : G \times A \quad \Gamma, x : A \vdash e' : G \times y B \\
\Gamma \vdash \text{do } x \leftarrow e; e' : G \times (x \cdot y) B \\
\Gamma \vdash e : A \\
\Gamma \vdash \text{return } e : G \times I A
\]

... for refining semantics

**Humble state**
\[
\text{State } A = \text{Store}(\mathcal{L}) \to A \times \text{Store}(\mathcal{L}) \\
\text{get : } (l : A \in \mathcal{L}) \to \text{State } A \\
\text{put : } (l : A \in \mathcal{L}) \to A \to \text{State } 1
\]

**vs. graded**
\[
\text{State } x A = \text{Store}(\text{reads}(x)) \to A \times \text{Store}(\text{writes}(x)) \\
\text{get : } (l : A \in \mathcal{L}) \to \text{State } \{r(l)\} A \\
\text{put : } (l : A \in \mathcal{L}) \to A \to \text{State } \{w(l)\} 1
\]
Graded monads

\[ G : \mathcal{E} \to [\mathbb{C}, \mathbb{C}] \]

\((\mathcal{E}, \cdot, I)\)

Functor

(Discrete) monoidal category

\[ G(x \cdot y) \]

\[ G(x \cdot I) \]

\[ \mu_{x,y} \]

\[ \eta \]

\[ \text{multiplication} \]

\[ \text{unit} \]

+ associativity and unitality axioms
(unordered) Graded monads

\[ G : \mathcal{E} \to [\mathbb{C}, \mathbb{C}] \]

(\mathcal{E}, \cdot, I)

Functor

(Discrete) monoidal category

\[ G(x \cdot y) \]

\[ G(I) \]

\[ \mu_{x,y} \]

\[ \eta \]

multiplication

unit

+ associativity and unitality axioms
Graded monads

\[ G : \mathbb{E} \to [\mathbb{C}, \mathbb{C}] \]

Functor

\((\mathbb{E}, \cdot, I, \leq)\)

Strict monoidal category

\[ G x \cdot G y \]

\[ \mu_{x,y} \]

\[ G(x \cdot y) \]

multiplication

+ associativity and unitality axioms

\[ \text{Id} \]

unit

\[ G I \]

approximation

+ monotonicity

\[ G x \]

+ \[ G(h : x \leq y) \]

unordered

graded monads
### Parametrised monads

\[ P : \mathbb{I}^{\text{op}} \times \mathbb{I} \rightarrow [\mathbb{C}, \mathbb{C}] \]

Functor

- **Associativity** and **Unitality** axioms

\[ P(i,j) \circ P(j,k) = P(i,k) \]

\[ \mu_{i,j,k} \]

\[ P(i,k) \]

**Multiplication**

\[ \eta_i \]

\[ P(i,i) \]

**Unit**

+ **Associativity** and **Unitality** axioms

[Atkey'06-'09 - Parameterised notions of computation]
Parametrised monads

\[ P : \mathbb{I}^{\text{op}} \times \mathbb{I} \rightarrow [\mathbb{C}, \mathbb{C}] \]

Functor

\[ P(i, j) \circ P(j, k) \]

\[ \mu_{i, j, k} \]

\[ P(i, k) \]

\[ \text{multiplication} \]

\[ \text{associativity and unitality axioms} \]

\[ \eta_i \]

\[ P(i, i) \]

\[ \text{unit} \]

\[ \{i\} \mathbin{C} \{j\} \]

\[ \{j\} \mathbin{C'} \{k\} \]

\[ \{i\} \mathbin{C; C'} \{k\} \]

\[ A \xrightarrow{f} P(i, j) B \]

\[ B \xrightarrow{g} P(j, k) C \]

\[ A \xrightarrow{\mu_{i, j, k} \circ Pg \circ f} P(i, k) C \]

cf Floyd-Hoare logic

[Atkey'06/'09 - Parameterised notions of computation]
Parametrised monads

\[ P : \mathcal{C}^{\text{op}} \times \mathcal{C} \to [\mathcal{C}, \mathcal{C}] \]

Functor

\[
\begin{align*}
P(i, j) \circ P(j, k) & \quad \text{Id} & \quad P(i, j) \\
\mu_{i,j,k} & \quad \eta_i & \quad P(f : i' \to i, g : j \to j') \\
P(i, k) & \quad P(i, i) & \quad P(i', j') \\
\text{multiplication} & \quad \text{unit} & \quad \text{approximation}
\end{align*}
\]

+ associativity and unitality axioms

+ dinaturality axioms

[Atkey'06/'09 - Parameterised notions of computation]
Can we unify their definitions?
Roadmap

- 2-category-graded monads
- + generalised unit

- unordered
- graded monads
- "oidify"

- category-graded monads

- discrete
- parameterised monads

- 2-category-graded monads
- + generalised unit

- monads
“Oidification”

1. concept shown to be equivalent to a single-object category

2. generalise that to a category with more than one object/morphism
Monads are **lax functors**

(Benabou 1967)

One object \(*\)

One morphism

\(\text{id} : * \to *\)

\[ T : 1 \to \text{Endo}(\mathbb{C}) \]

Recall: functor axioms

\[ F : \mathbb{C} \to \mathbb{D} \]

\[ \text{id} = F \text{id} \]

\[ Fg \circ Ff = F(g \circ f) \]

Lax functor axioms

\[ \text{id} \Rightarrow F \text{id} \]

\[ Fg \circ Ff \Rightarrow F(g \circ f) \]

+ associativity/unitality

\(\mathbb{D}\) is a 2-category

Here natural transformations

\[ \eta : \text{id} \Rightarrow T \text{id} \]

\[ \mu : T \text{id} \circ T \text{id} \Rightarrow T \text{id} \]
Oidifying a monad

Monad

1

\[
T \downarrow
\]

Endo(\mathbb{C})

\text{oidification} / \text{horizontal categorification}

\text{“Category-graded monad”}

\[
\mathcal{F}^{\text{op}}
\]

\[
\downarrow T
\]

Endo(\mathbb{C})

\text{Lax functor}

\eta : \text{Id} \Rightarrow T \text{id}

\mu : T \text{id} \circ T \text{id} \Rightarrow T \text{id}

\eta_x : \text{Id} \Rightarrow T \text{id}_x

\mu_{f,g} : Tf \circ Tg \Rightarrow T(g \circ f)

\text{(Benabou 1967)}
1. Unordered graded monads are category-graded monads

"Category-graded monad"

\[ \mathcal{F}^{\text{op}} \]

\[ T \]

\[ \text{Endo}(\mathcal{C}) \]

\[ \eta_x : \text{Id} \Rightarrow T\text{id}_x \]

\[ \mu_{f,g} : Tf \circ Tg \Rightarrow T(g \circ f) \]

Monoid-graded monad \( G \)

\[ (\mathbb{E}, \bullet, I) \]

\[ G \]

\[ [\mathcal{C}, \mathcal{C}] \]
Monoids are one-object categories are discrete monoidal categories
1. Unordered graded monads are category-graded monads

Category-graded monad

\[ \mathcal{F}^{\text{op}} \]
\[ \downarrow T \]
\[ \text{Endo}(\mathbb{C}) \]

\[ \eta_x : \text{Id} \Rightarrow T\text{id}_x \]
\[ \mu_{f,g} : Tf \circ Tg \Rightarrow T(g \circ f) \]

Monoid-graded monad \( G \)

\[ 1_{(\mathbb{E}^{\text{op}}, \cdot, I)} \equiv (\mathbb{E}, \cdot, I) \]
\[ \downarrow \quad \Downarrow \quad \Downarrow \]
\[ \text{Endo}(\mathbb{C}) \equiv [\mathbb{C}, \mathbb{C}] \]

\[ \eta_\star : \text{Id} \Rightarrow G\text{I} \]
\[ \mu_{x,y} : Gx \circ Gy \Rightarrow G(x \cdot y) \]
2. Graded monads are 2-category-graded monads

**2-category-graded monad**

Category-graded monad

\[ T : \mathcal{F}^{\text{op}} \to \text{Endo}(\mathbb{C}) \]

\[ \eta_x : \text{Id} \Rightarrow T \text{id}_x \]

\[ \mu_{f,g} : Tf \circ Tg \Rightarrow T(g \circ f) \]

+ 2-morphism mapping

\[ T(h : g \Rightarrow f) : Tg \Rightarrow Tf \]

(i.e., \( \mathcal{F} \) is a 2-category)
Pomonoids are one object 2-categories are monoidal categories
2. Graded monads are 2-category-graded monads

**2-category-graded monad**

\[ T : \mathcal{F}^{\text{op}} \to \text{Endo}(\mathbb{C}) \]

\[ \eta_x : \text{Id} \Rightarrow T \text{id}_x \]

\[ \mu_{f,g} : Tf \cdot Tg \Rightarrow T(g \circ f) \]

\[ T(h : g \Rightarrow f) : Tg \Rightarrow Tf \]

**2-morphism mapping**

**(Ordered) Graded monad**

\[ G : 1_{(\mathcal{E}^{\text{op}}, \cdot, I, \leq)} \to \text{Endo}(\mathbb{C}) \]

\[ \eta_x : \text{Id} \Rightarrow GI \]

\[ \mu_{x,y} : Gx \cdot Gy \Rightarrow G(x \cdot y) \]

\[ G(h : x \leq y) : Gx \Rightarrow Gy \]
Roadmap

2-category-graded monads

unordered

graded monads

category-graded monads

“oidify”

category-graded monads + generalised unit

discrete

parameterised monads

monads
3. Discrete parameterised monads are category-graded monads

\[ P : \mathbb{J}^{op} \times \mathbb{J} \rightarrow [\mathbb{C}, \mathbb{C}] \]
where \( \mathbb{J} \) has only identity morphisms

Define the category of \( \mathbb{J} \)-“dominoes”

\[ \nabla(\mathbb{J})_1 = \mid \mathbb{J} \mid \times \mid \mathbb{J} \mid \]

(e.g., composition \( (j, k) \circ (i, j) = (i, k) \))

Define a category graded monad

\[ T : \nabla(\mathbb{J}) \rightarrow \text{Endo}(\mathbb{C}) \]
with \( T(i, j) = P(i, j) \)

\[ \eta_i : \text{Id} \Rightarrow T(i, i) = \eta_i^P \]

\[ \mu_{(i,j),(j,k)} : T(i,j) \circ T(j,k) \Rightarrow T(i,k) = \mu_{i,j,k}^P \]
Parameterised monads have some extra structure

\[ P(i, j) \xrightarrow{P(f : i' \to i, g : j \to j')} P(i', j') \]

morphism mapping (approximation) + dinaturality axioms
Generalised units

arises from lax natural transformations (Street, 1972)

Wide sub-category

\[ \mathcal{S} \subseteq \mathcal{F} \]

Family of morphisms

\[ \hat{\eta}_f : X \to Y \in \mathcal{S} : \text{Id} \to Tf \]

4. Parameterised monads are category-graded monads + \( \hat{\eta} \)

Paper shows details
Roadmap

2-category-graded monads
+ generalised unit

unordered
graded monads

category-graded monads

“oidify”

discrete
parameterised monads
Example

| $\mathcal{F}$ | = \{free, critical\}  
lock : free $\rightarrow$ critical  
unlock : critical $\rightarrow$ free  
get, put : critical $\rightarrow$ critical

ConcSt : $\mathcal{F}^{op}$ $\rightarrow$ Endo($\mathbb{C}$)

get : ConcSt $\text{get } S$
put : $S$ $\rightarrow$ ConcSt $\text{put } 1$
lock : ConcSt $\text{lock } 1$
unlock : ConcSt $\text{unlock } 1$

spawn : ($\forall f$. ConcSt ($f : \text{free } \rightarrow \text{free}$) 1) $\rightarrow$ ConcSt $f$ 1
Conclusions

- Shows us where graded & parameterised overlap
- A more general structure that captures both aspects: tracing + restriction

Category-graded monad

\[ \mathcal{F}^\text{op} \]

\[ \begin{array}{c}
\downarrow T \\
\text{Endo}(\mathbb{C})
\end{array} \]

\[ \eta_x : \text{Id} \Rightarrow T id_x \]

\[ \mu_{f,g} : Tf \circ Tg \Rightarrow T(g \circ f) \]

Thank you!

\[ \text{granule-project.github.io} \]