Degrading lists

Dylan McDermott
Reykjavik University

Maciej Piróg
Wrocław University

Tarmo Uustalu
Reykjavik University
Tallinn University of Technology

PPDP 2020

What is the relationship between monads and graded monads?

- Monads T organize computations into sets TX
  (e.g. TX = lists over X)
- Graded monads organize computations into sets $T_g X$
  (e.g. $T_g X$ = lists over X of length g)
- The grades g provide quantitative information
  (e.g. number of alternatives in a nondeterministic computation)

What is the relationship between monads and graded monads?

- Monads T organize computations into sets TX (e.g. TX = lists over X)
- Graded monads organize computations into sets $T_g X$ (e.g. $T_g X$ = lists over X of length g)
- The grades g provide quantitative information (e.g. number of alternatives in a nondeterministic computation)

Specifically: can we construct monads from graded monads?

Monads and graded monads

A monad consists of

- A functor $T : \mathbf{Set} \to \mathbf{Set}$
  (with $\mathrm{map}\, f : TX \to TY$ for each $f : X \to Y$)
- A unit $\eta_X : X \to TX$ for each X (aka `return`)
- A multiplication $\mu_X : T(TX) \to TX$ for each X (aka `join`)

Example (non-empty lists):

$$TX = \mathrm{List}_+ X \qquad \eta\, x = [x] \qquad \mu\, \mathrm{xss} = \mathrm{concat}\, \mathrm{xss}$$

Alternatively:

- A set TX for each set X
- A unit $\mathrm{return} : X \to TX$ for each X
- A bind operator $\gg\!= : TX \to (X \to TY) \to TY$ for each X, Y

(in both cases, satisfying some laws)

Monads and graded monads

Given a monoid of grades:

$$(\mathcal{G}, \cdot, 1)$$

A $\mathcal{G}$-graded monad consists of

- A functor $T_g : \mathbf{Set} \to \mathbf{Set}$ for each grade $g \in \mathcal{G}$
  (with $\mathrm{map}\, f : T_g X \to T_g Y$ for each $f : X \to Y$)
- A unit $\eta_X : X \to T_1 X$ for each X
- A multiplication $\mu_{g,g',X} : T_g(T_{g'} X) \to T_{g \cdot g'} X$ for each g, g', X

(satisfying some laws)

Alternatively, use

$$\gg\!= : T_g X \to (X \to T_{g'} Y) \to T_{g \cdot g'} Y$$

Monads and graded monads

Given a monoid of grades:

$$(\mathcal{G}, \cdot, 1)$$

A $\mathcal{G}$-graded monad consists of

- A functor $T_g : \mathbf{Set} \to \mathbf{Set}$ for each grade $g \in \mathcal{G}$ (with $\mathrm{map}\, f : T_g X \to T_g Y$ for each $f : X \to Y$)
- A unit $\eta_X : X \to T_1 X$ for each X
- A multiplication $\mu_{g,g',X} : T_g(T_{g'} X) \to T_{g \cdot g'} X$ for each g, g', X

(satisfying some laws)

Example (non-empty lists)

- Grades are positive integers with multiplication $(\mathbb{N}_+, \cdot, 1)$
- Graded monad is:

$$T_n X = \mathrm{List}_{+=n} X \qquad \eta\, x = [x] \qquad \mu\, \mathrm{xss} = \mathrm{concat}\, \mathrm{xss}$$

Monads and graded monads

Given a monoid of grades:

$$(\mathcal{G}, \cdot, 1)$$

A $\mathcal{G}$-graded monad consists of

- A functor $T_g : \mathbf{Set} \to \mathbf{Set}$ for each grade $g \in \mathcal{G}$
  (with $\mathrm{map}\, f : T_g X \to T_g Y$ for each $f : X \to Y$)
- A unit $\eta_X : X \to T_1 X$ for each X
- A multiplication $\mu_{g,g',X} : T_g(T_{g'} X) \to T_{g \cdot g'} X$ for each g, g', X

(satisfying some laws)

Example (possibly-empty lists)

- Grades are natural numbers with multiplication $(\mathbb{N}, \cdot, 1)$
- Graded monad is:

$$T_n X = \mathrm{List}_{=n} X \qquad \eta\, x = [x] \qquad \mu\, \mathrm{xss} = \mathrm{concat}\, \mathrm{xss}$$

Monads from graded monads

Can we turn graded monads T into non-graded monads $\hat{T}$?

For example:

- Can we construct a monad by constructing the corresponding graded monad first?
  (e.g. [Fritz and Perrone '18]'s Kantorovich monad)
- If we can model a language with grades, can we model the language without grades?

$$\begin{array}{ccc} \vdash_g M : \mathbf{int} & \longmapsto & [\![M]\!] \in T_g\,\mathbb{Z} \\ \downarrow & & \downarrow \lambda_g \\ \vdash \underline{M} : \mathbf{int} & \longmapsto & [\![\underline{M}]\!] \in \hat{T}\,\mathbb{Z} \end{array}$$

- Do we have

$$\mathrm{List}_{+=} \mapsto \mathrm{List}_{+} \qquad \mathrm{List}_{=} \mapsto \mathrm{List}$$

Degradings

A degrading of a graded monad (T, η, μ) consists of

- A monad $(\hat{T}, \hat{\eta}, \hat{\mu})$
- Functions $\lambda_{g,X} : T_g X \to \hat{T} X$ preserving the structure, e.g. the multiplications:

$$\begin{tikzcd} T_g(T_{g'}X) \arrow[r, "\mu"] \arrow[d, "\lambda_g \circ \mathrm{map}\, \lambda_{g'}"'] & T_{g \cdot g'}X \arrow[d, "\lambda_{g \cdot g'}"] \\ \hat{T}(\hat{T}X) \arrow[r, "\hat{\mu}"'] & \hat{T}X \end{tikzcd}$$

Example: $(\mathrm{List}_+, [-], \mathrm{concat})$ forms a degrading of $(\mathrm{List}_{+=}, [-], \mathrm{concat})$

$$\lambda_{n,X} : \mathrm{List}_{+=n}X \subseteq \mathrm{List}_+X$$

Constructing degradings

Take the coproduct of $g \mapsto T_g$:

$$\hat{T} : \mathbf{Set} \longrightarrow \mathbf{Set} \qquad\qquad \lambda_g : T_g X \longrightarrow \hat{T} X$$

$$\hat{T}\, X = \textstyle\sum_{g \in \mathcal{G}} T_g X \qquad\qquad t \longmapsto (g, t)$$

so that elements of $\hat{T}X$ are pairs $(g \in \mathcal{G}, t \in T_g X)$

- Have a unit

$$\begin{aligned} \hat{\eta} : X &\longrightarrow \textstyle\sum_{g \in \mathcal{G}} T_g X \\ x &\longmapsto (1, \eta\, x) \end{aligned}$$

- But what about the multiplication?

$$\hat{\mu} : \textstyle\sum_{g \in \mathcal{G}} T_g \big(\sum_{g' \in \mathcal{G}} T_{g'} X \big) \xrightarrow{?} \sum_{g'' \in \mathcal{G}} T_{g''} X$$

  from

$$\mu_{g,g'} : T_g(T_{g'} X) \longrightarrow T_{g \cdot g'} X$$

Algebraic coproducts

The coproduct $\hat{T}$ is an algebraic coproduct if:

- It forms a degrading
- For every other degrading T', there are unique structure-preserving functions $\hat{T}X \to T'X$

(more generally: algebraic Kan extension)

For models of effectful languages:

- A computation would be a pair of a g and a computation of grade g
- For any other model given by a degrading T', the unique functions preserve interpretations of terms

Algebraic coproducts

Algebraic Kan extensions sometimes exist:

Fritz and Perrone, A Criterion for Kan Extensions of Lax Monoidal Functors

but often don't

- Neither $\mathrm{List}_{+=}$ nor $\mathrm{List}_{=}$ has an algebraic coproduct

Algebraic coproducts

Algebraic Kan extensions sometimes exist:

Fritz and Perrone, A Criterion for Kan Extensions of Lax Monoidal Functors

but often don't

- Neither $\mathrm{List}_{+=}$ nor $\mathrm{List}_{=}$ has an algebraic coproduct

Introduce two weakenings:

- Unique shallow degrading: don't require structure-preservation for $\hat{T}X \to T'X$
- Initial degrading: don't require a coproduct

Algebraic coproduct $\iff$ unique shallow degrading $\wedge$ initial degrading

First weakening: unique shallow degrading

If the coproduct $\hat{T}$ uniquely forms a degrading, call it the unique shallow degrading

- There are unique λ-preserving functions $\hat{T}X \to T'X$, but they don't preserve all of the structure

Non-example

List does not form the unique shallow degrading of $\mathrm{List}_{=}$

$$\hat{\mu}\,\mathrm{xss} = \mathrm{concat}\ \mathrm{xss} \qquad \text{or} \qquad \hat{\mu}\,\mathrm{xss} = \begin{cases} [] & \text{if } [] \in \mathrm{xss} \\ \mathrm{concat}\ \mathrm{xss} & \text{otherwise} \end{cases}$$

Example

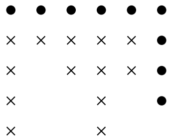
$(\mathrm{List}_{+}, [-], \mathrm{concat})$ is the unique shallow degrading of $\mathrm{List}_{+=}$

How many list monads are there?

Answer: infinitely many

$$T = \mathrm{List}_{+} \qquad \eta\, x = [x]$$

$$\mu\, [\mathrm{xs}_1, \ldots, \mathrm{xs}_n] \;=\; \mathrm{head}\, \mathrm{xs}_1 :: \cdots :: \mathrm{head}\, \mathrm{xs}_{n-1} :: \mathrm{xs}_n$$



How many list monads are there?

Answer: infinitely many

$$T = \mathrm{List}_{+} \qquad \eta\, x = [x]$$

$$\mu\, \mathrm{xss} \;=\; \begin{cases} \mathrm{concat\,xss} & \text{if xss is a singleton, or all-singletons} \\ \mathrm{take}\ 11\ (\mathrm{concat\,xss}) & \text{otherwise} \end{cases}$$

$$\begin{array}{cccccc}
\bullet & \bullet & \bullet & \bullet & \times & \times \\
\bullet & \bullet & \bullet & \times & \times & \times \\
\bullet & & \bullet & \times & \times & \times \\
\bullet & & & \times & & \times \\
\bullet & & & \times & &
\end{array}$$

How many list monads are there?

Answer: infinitely many

$$T = \mathrm{List}_{+} \qquad \eta\, x = [x, x]$$

$$\mu\, \mathrm{xss} \;=\; \mathrm{head}\,(\mathrm{head}\,\mathrm{xss}) :: \mathrm{concat}(\mathrm{tail}\,(\mathrm{map}\,\mathrm{tail}\,\mathrm{xss}))$$

$$\begin{array}{cccccc}
\bullet & \times & \times & \times & \times & \times \\
\times & \bullet & \bullet & \bullet & \bullet & \bullet \\
\times & & \bullet & \bullet & \bullet & \bullet \\
\times & & & \bullet & & \bullet \\
\times & & & \bullet & &
\end{array}$$

How many list monads are there?

Answer: infinitely many (for both non-empty and possibly-empty)

- Can discard elements
- Can duplicate elements
- Can have no finite presentation
- Can have $\eta\, x \neq [x]$

How many list monads are there?

Answer: infinitely many (for both non-empty and possibly-empty)

- Can discard elements
- Can duplicate elements
- Can have no finite presentation
- Can have $\eta\, x \neq [x]$

But for List_{+}: only one agrees with the graded monad

List_+ is a unique shallow degrading

If a non-empty list monad satisfies

$$\mu\,\mathrm{xss} = \mathrm{concat}\,\mathrm{xss} \qquad \text{(for balanced xss)}$$

then $\mu = \mathrm{concat}$

Proof sketch:

1. Show that $\mu\,\mathrm{xss}$ cannot discard elements, by considering elements of $\mathrm{List}_+^3 X$
2. Implies μ cannot duplicate elements
3. Prove $\mu[[x, y], [z]] = [x, y, z] = \mu[[x], [y, z]]$ by brute force
4. So μ just concatenates, then permutes the result based on the length
5. These permutations must be identities

Second weakening: initial degrading

$\hat{T}$ is the initial degrading of a graded monad T if:

- It is a degrading
- For any other degrading T', there are unique structure-preserving functions

$$\hat{T}X \to T'X$$

But: $\hat{T}$ does not have to be the coproduct
(it is actually a Kan extension in $\mathbf{MonCat}$ instead of $\mathbf{Cat}$)

Constructing initial degradings

Start with a graded monad T

1. Take the (ordinary) coproduct of $g \mapsto T_g$
2. Construct the free monad on the coproduct
3. Quotient to get a degrading

These often exist, but are not intuitive:

- $\mathrm{List}_{=}$ and $\mathrm{List}_{+=}$ have initial degradings
- They don't have simple descriptions: they are not List or List_{+}

Conclusions

Degradings are much more complicated than they first seem

- List_{+} is the unique shallow degrading, but not the initial degrading, of $\mathrm{List}_{+=}$
- List isn't the unique shallow degrading or the initial degrading of $\mathrm{List}_{=}$

Neither is an algebraic coproduct

There are a lot of list monads:

`https://github.com/maciejpirog/exotic-list-monads`