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 $T_gX$ (e.g. $T_gX =$ 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. $TX = \text{lists over } X$)
- Graded monads organize computations into sets $T_gX$  
  (e.g. $T_gX = \text{lists over } X$ 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 \( T : \text{Set} \to \text{Set} \)
  (with 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):

\[
TX = \text{List}_+X \quad \eta x = [x] \quad \mu xss = \text{concat xss}
\]

Alternatively:

- A set \( TX \) for each set \( X \)
- A unit 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:

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

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

- A functor $T_g : \text{Set} \to \text{Set}$ for each grade $g \in \mathbb{G}$
  (with 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

$$\ggg : T_g X \to (X \to T_{g'} Y) \to T_{g \cdot g'} Y$$
Monads and graded monads

Given a monoid of grades:

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

A \(\mathcal{G}\)-graded monad consists of

▶ A functor \(T_g : \text{Set} \rightarrow \text{Set}\) for each grade \(g \in \mathcal{G}\)
  (with map \(f : T_gX \rightarrow T_gY\) for each \(f : X \rightarrow Y\))
▶ A unit \(\eta_X : X \rightarrow T_1X\) for each \(X\)
▶ A multiplication \(\mu_{g,g',X} : T_g(T_g'X) \rightarrow T_{g \cdot g'}X\) 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:

\[ T_nX = \text{List}_{+ = n}X \quad \eta x = [x] \quad \mu xss = \text{concat } xss \]
Monads and graded monads

Given a monoid of grades:

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

A \(\mathcal{G}\)-graded monad consists of

▶ A functor \(T_g : \text{Set} \to \text{Set}\) for each grade \(g \in \mathcal{G}\)
  (with 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:

\[
T_n X = \text{List}_{\equiv n} X \quad \eta x = [x] \quad \mu xss = \text{concat } xss
\]
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?

\[
\begin{align*}
\vdash_g M : \text{int} & \quad \longmapsto \quad [M] \in T_g \mathbb{Z} \\
\downarrow & \quad \downarrow \lambda_g \\
\vdash M : \text{int} & \quad \longmapsto \quad [M] \in \hat{T} \mathbb{Z}
\end{align*}
\]

- Do we have

\[
\text{List}_{+\equiv} \mapsto \text{List}_+ \quad \text{List}_{\equiv} \mapsto \text{List}
\]
Degradings

A degrading of a graded monad \((T, \eta, \mu)\) consists of

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

\[
\begin{align*}
T_g(T_{g'}X) & \xrightarrow{\mu} T_{g \cdot g'}X \\
\lambda_g \circ \text{map} \lambda_{g'} & \downarrow \quad \downarrow \lambda_{g \cdot g'} \\
\hat{T}(\hat{T}X) & \xrightarrow{\hat{\mu}} \hat{T}X
\end{align*}
\]

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

\[
\lambda_{n,X} : \text{List}_{+=n}X \subseteq \text{List}_+X
\]
Constructing degradings

Take the coproduct of \( g \mapsto T_g \):

\[
\hat{T} : \text{Set} \to \text{Set} \\
\hat{T} X = \sum_{g \in G} T_g X \\
\overset{\lambda}{\to} T_g X \overset{t}{\mapsto} (g, t)
\]

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

- Have a unit

\[
\hat{\eta} : X \to \sum_{g \in G} T_g X \\
x \mapsto (1, \eta x)
\]

- But what about the multiplication?

\[
\hat{\mu} : \sum_{g \in G} T_g \left( \sum_{g' \in G} T_{g'} X \right) \overset{?}{\to} \sum_{g'' \in G} T_{g''} X
\]

from

\[
\mu_{g,g'} : T_g (T_{g'} X) \to T_{g \cdot g'} X
\]
The coproduct $\hat{T}$ is an algebraic coproduct if:

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

(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 $\text{List}_+=\text{List}_-$ nor $\text{List}_-$ 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 $\text{List}_{\leq}$ nor $\text{List}_{\geq}$ 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 $\land$ 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 $\lambda$-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 $\text{List}_-$

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

**Example**

$(\text{List}_+, [-], \text{concat})$ is the unique shallow degrading of $\text{List}_{+-}$
How many list monads are there?

Answer: infinitely many

\[
T = \text{List}_+ \quad \eta x = [x] \\
\mu [xs_1, \ldots, xs_n] = \text{head} \, xs_1 :: \cdots :: \text{head} \, xs_{n-1} :: xs_n
\]
How many list monads are there?

Answer: infinitely many

\[ T = \text{List}_+ \quad \eta x = [x] \]

\[ \mu \text{xss} = \begin{cases} \text{concat xss} & \text{if xss is a singleton, or all-singletons} \\ \text{take 11 (concat xss)} & \text{otherwise} \end{cases} \]
How many list monads are there?

Answer: infinitely many

\[ T = \text{List}_+ \quad \eta x = [x, x] \]

\[ \mu \text{xss} = \text{head} \left( \text{head} \text{xss} \right) :: \text{concat} \left( \text{tail} \left( \text{map} \text{tail} \text{xss} \right) \right) \]
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

\[ \mu xss = \text{concat } xss \]  
(for balanced \(xss\))

then \(\mu = \text{concat} \)

Proof sketch:

1. Show that \(\mu xss\) cannot discard elements, by considering elements of \(\text{List}_3^X\)
2. Implies \(\mu\) cannot duplicate elements
3. Prove \(\mu[[x, y], [z]] = [x, y, z] = \mu[[x], [y, z]]\) by brute force
4. So \(\mu\) 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

\[
\hat{T}X \rightarrow T'X
\]

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

- $\mathsf{List}_=$ and $\mathsf{List}_{+=}$ have initial degradings
- They don’t have simple descriptions: they are not $\mathsf{List}$ or $\mathsf{List}_+$
Conclusions

Degradings are much more complicated than they first seem

▶ List$_+$ is the unique shallow degrading, but not the initial degrading, of List$_+\leq$

▶ List isn’t the unique shallow degrading or the initial degrading of List$_\leq$

Neither is an algebraic coproduct

There are a lot of list monads:

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