Counting Monads on Lists

Dylan McDermott, **Maciej Piróg**, Tarmo Uustalu

CLA 2023

Combinatorial approach to category theory (monads in particular)

The infamous definition:

A monad is a monoid in the category of endofunctors

Problem: Given an endofunctor, what monoid structures are there?

Motivation: e.g., composition of monads

This talk

- We focus on the **list** endofunctor on Set.
- Work in progress. Some known results, some new results, some directions to go from here.
- The main new result:

How many list monads are there?

Lists (finite sequences)

- LA – set of all lists with elements coming from the set A
- $[x_1, \ldots, x_n]$ – constructing lists by enumerating elements
- $xs +\!\!+\, ys$ – concatenating lists (e.g., $[1,2] +\!\!+ [3,4,5] = [1,2,3,4,5]$)
- $xs \in LA$, $xss \in L(LA)$, $xsss \in L(L(LA))$ – naming convention for lists
- $Lf([x_1, \ldots, x_n] = [f(x_1), \ldots, f(x_n)]$ – "map"

Monads on lists

Two families of functions indexed by sets: $\eta_A : A \to LA$ and $\mu_A : L(LA) \to LA$

$$\boxed{\text{ASSOC}} \quad \mu_A \circ L\mu_A = \mu_A \circ \mu_{LA} : L(L(LA)) \to LA$$

$$\boxed{\text{L-UNIT}} \quad \mu_A \circ \eta_{LA} = \mathrm{id} : LA \to LA$$

$$\boxed{\text{R-UNIT}} \quad \mu_A \circ L\eta_A = \mathrm{id} : LA \to LA$$

$$\boxed{\eta\text{-NATURAL}} \quad \eta_B \circ f = Lf \circ \eta_A : A \to LB \quad \text{for all } f : A \to B$$

$$\boxed{\mu\text{-NATURAL}} \quad \mu_B \circ L(Lf) = Lf \circ \mu_A : L(LA) \to LB \quad \text{for all } f : A \to B$$

“The” list monad

$$\eta(x) = [x]$$
$$\mu([xs, \ldots, zs]) = xs \mathbin{+\!\!+} \cdots \mathbin{+\!\!+} zs$$

E.g., ASSOC:

$$\mu(\mu([[[1],[2,3]],[[4],[],[5,6]]])) = \mu([[1],[2,3],[],[4],[5,6]]) = [1,2,3,4,5,6]$$
$$\mu(L\mu([[[1],[2,3]],[[4],[],[5,6]]])) = \mu([[1,2,3],[4,5,6]]) = [1,2,3,4,5,6]$$

But... did I just say families indexed by **sets**...?!?

The naturality rules η-NATURAL and μ-NATURAL give us that to define a list monad it is enough to define $\eta_{\mathbb{N}}$ and $\mu_{\mathbb{N}}$.

Intuitively: η and μ cannot "look" at what the particular element is.

Are there other monads on lists?

The "global error" monad

$$\begin{aligned} \eta(x) &= [x] \\ \mu([xs_1, \ldots, xs_n]) &= [\,] \quad \text{if } xs_k \text{ empty for any } k \\ \mu([xs_1, \ldots, xs_n]) &= xs_1 \mathbin{+\!\!+} \cdots \mathbin{+\!\!+} xs_n \quad \text{otherwise} \end{aligned}$$

(see our PPDP 2020 paper or the `exotic-list-monads` Haskell library)

Are there other monads on lists?

The "mini" monad

$$
\begin{aligned}
\eta(x) &= [x] \\
\mu([xs]) &= xs \\
\mu([[x], \ldots, [z]]) &= [x, \ldots, z] \\
\mu(xs) &= [\,] \quad \text{otherwise}
\end{aligned}
$$

(see our PPDP 2020 paper or the `exotic-list-monads` Haskell library)

Are there other monads on lists?

The "maze walk" monad

$$\eta(x) = [x]$$

$$\mu([xs_1, \ldots, xs_n]) = [\,] \quad \text{if } xs_k \text{ empty for any } k$$

$$\mu([xs_1, \ldots, xs_n]) = p(xs_1) \mathbin{+\!\!+} \cdots \mathbin{+\!\!+} p(xs_{n-1}) \mathbin{+\!\!+} xs_n \quad \text{otherwise}$$

$$\text{where } p([x_1, \ldots, x_m]) = [x_1, \ldots, x_{m-1}, x_m, x_{m-1}, \ldots, x_1]$$

(see our PPDP 2020 paper or the `exotic-list-monads` Haskell library)

Are there other monads on lists?

The "stutter" monad

For any natural number n, in Haskell:

```
join xss | null xss
         = []
         | any (not . isSingle) (init xss) || null (last xss)
         = replicateLast (n + 1) (concat $
                                   takeWhile isSingle (init xss))
         | otherwise
         = concat xss
```

(see our PPDP 2020 paper or the `exotic-list-monads` Haskell library)

Are there other monads on lists?

Previous results (PPDP 2020):

- There are infinitely many list monads
- They can have rather complicated definitions
- μ can discard, duplicate, and shuffle elements of lists
- Infinitely many list monads arise from finite equational theories
- Some list monads do not arise from any finite equational theory

Are there other monads on lists?

Previous results (PPDP 2020):

- There are infinitely many list monads
- They can have rather complicated definitions
- μ can discard, duplicate, and shuffle elements of lists
- Infinitely many list monads arise from finite equational theories
- Some list monads do not arise from any finite equational theory

How many exactly?

How many list monads are there?

- There are at least $\aleph_0$ list monads
- Every list monad is uniquely characterised by $\mu_{\mathbb{N}} : L(L\mathbb{N}) \to L\mathbb{N}$ and $\eta_{\mathbb{N}} : \mathbb{N} \to L\mathbb{N}$, so there are at most $2^{\aleph_0}$ list monads.

So, can we construct an uncountable family of list monads?

CORE list monads

CORE = Concatenate OR Error

- $\eta(x) = [x]$
- $\mu([xs_1, \ldots, xs_n])$ is either empty or equal to $xs_1 \mathbin{+\!\!+} \cdots \mathbin{+\!\!+} xs_n$

This simplifies definition to specifying which lists of lists are not mapped to the empty list.

Attempt 1: "Good" sets

Each monad is defined by a property that is preserved by concatenation of an appropriate number of elements:

*We call a set $C \subseteq \mathbb{N}$ **good** if $0 \notin C$, $1 \in C$,*

and for all $k \in C$ and $n_1, \ldots, n_k \in C$ it is the case that $\sum_{i=1}^{k} n_i \in C$.

Examples:
$\{1\}$
$\{n \mid n \text{ is odd}\}$
$\{1\} \cup \{n, n+1, \ldots\}$ for any $n > 1$

Attempt 1: "Good" sets

We call a set $C \subseteq \mathbb{N}$ ***good*** *if $0 \notin C$, $1 \in C$,*
and for all $k \in C$ and $n_1, \ldots, n_k \in C$ it is the case that $\sum_{i=1}^{k} n_i \in C$.

Theorem: Every good set C induces a monad with $\eta(a) = [a]$ and

$$
\begin{aligned}
\mu([xs]) &= xs \\
\mu([[x_1], \ldots, [x_n]]) &= [x_1, \ldots, x_n] \\
\mu([xs_1, \ldots, xs_k]) &= xs_1 \mathbin{+\!\!+} \cdots \mathbin{+\!\!+} xs_k && \text{if } k \in C \text{ and } |xs_i| \in C \text{ for all } i = 1, \ldots, k \\
\mu(xss) &= [\,] && \text{otherwise}
\end{aligned}
$$

How many “good” sets are there?

*We call a set $C \subseteq \mathbb{N}$ **good** if $0 \notin C$, $1 \in C$,*
and for all $k \in C$ and $n_1, \ldots, n_k \in C$ it is the case that $\sum_{i=1}^{k} n_i \in C$.

Theorem: Let $C^- = \{n - 1 \mid n \in C\}$. Then, C is good if and only if C^- is a numerical monoid, that is, $0 \in C^-$ and for all $k, n \in C^-$ it is the case that $k + n \in C^-$.

It is a known fact that there are only $\aleph_0$ numerical monoids.

Attempt 2: Uncountably many list monads

Let G be a subset of the set of odd natural numbers.

We define a monad with $\eta(a) = [a]$ and

$$\begin{aligned}
\mu([xs]) &= xs \\
\mu([[x_1], \ldots, [x_n]]) &= [x_1, \ldots, x_n] \\
\mu([[x_1, \ldots, x_n], [y]]) &= [x_1, \ldots, x_n, y] \quad \text{if } n \in G \\
\mu(xss) &= [\,] \quad \text{otherwise}
\end{aligned}$$

Open questions and hypotheses

- Just knowing the cardinality of the set of list monads is not enough. Is some form of classification/characterisation theorem possible for (CORE) list monads?
- Hypothesis: There is no list monad with $\eta(x) \neq [x]$. (We know there is such a monad on non-empty lists.)

Why is this difficult?

- Problem: Each list monad is an infinite object.
- Problem: Working with lists of lists of lists has high mental complexity.
- Desired solution: employ some non-elementary techniques.

Other functors

- Fact (PPDP'20): Every list monad induces a monad on non-empty lists by the Id $\times$ - construction.
- Corollary: There are $2^{\aleph_0}$ monads on non-empty lists.
- Hypothesis: We can freely adjoin "global error" to a non-empty list monad to obtain a list monad – amazingly, this construction seems to work exactly for monads that do not discard elements.
- Hypothesis: The construction seems to extend to monads on multisets.