(Co)Monads

Alyson Mei

December 20, 2025

1 Introduction

Conventions. Throughout this text, we adopt the following conventions:

– All categories are assumed to be locally small.

– Standard polymorphic notation is used freely when no ambiguity arises.

– All diagrams are assumed to commute unless stated otherwise.

2 Monads

Definition 2.1 (Monoid object)**.** Let $(\mathcal{C}, \otimes, I)$ be a monoidal category. A monoid object in $\mathcal{C}$ consists of:

- an object $M \in \mathrm{obj}(\mathcal{C})$,
- a multiplication morphism
$$\mu : M \otimes M \to M,$$
- a unit morphism
$$\eta : I \to M,$$

satisfying the unit and associativity diagrams:

$$\begin{tikzcd}
M \arrow[r, "M\otimes\eta"] \arrow[dr, "\mathrm{id}"'] & M \otimes M \arrow[d, dashed, "\mu"] & M \arrow[l, "\eta\otimes M"'] \arrow[dl, "\mathrm{id}"] \\
& M &
\end{tikzcd}
\qquad
\begin{tikzcd}
M^{\otimes 3} \arrow[r, "\mu\otimes M"] \arrow[d, "M\otimes\mu"'] & M^{\otimes 2} \arrow[d, "\mu"] \\
M^{\otimes 2} \arrow[r, "\mu"'] & M
\end{tikzcd}.$$

Example 2.2.

– Categories with finite products $(\mathcal{C}, \times, 1)$ are monoidal:

  ◦ In $\mathcal{S}et$, monoid objects are ordinary monoids;

  ◦ In $\mathrm{Vect}_{\mathbf{k}}$, monoid objects are associative unital algebras over a field $\mathbf{k}$;

– The category of endofunctors $([\mathcal{C}, \mathcal{C}], \circ, \mathrm{id})$ is monoidal; monoid objects in this category are called monads.

Definition 2.3 (Monad)**.** Monad is a triple

$$(M : \mathcal{C} \to \mathcal{C}, \eta : \mathrm{id}_{\mathcal{C}} \to M, \mu : M^2 \to M),$$

satisfying the unit and associativity diagrams:

$$\begin{tikzcd}
M \arrow[r, "\eta\circ M"] \arrow[dr, "\mathrm{id}_M"'] & M^2 \arrow[d, "\mu"] & M \arrow[l, "M\circ\eta"'] \arrow[dl, "\mathrm{id}_M"] \\
& M &
\end{tikzcd}
\qquad
\begin{tikzcd}
M^3 \arrow[r, "M\circ\mu"] \arrow[d, "\mu\circ M"'] & M^2 \arrow[d, "\mu"] \\
M^2 \arrow[r, "\mu"'] & M
\end{tikzcd}.$$

Example 2.4 (List monad)**.** Consider the functor List : $\mathcal{S}et \to \mathcal{S}et$, which sends a set to the set of all finite lists of its elements:

$$\begin{tikzcd}
A \arrow[d, "f"] \\
B
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathrm{List}\, A \arrow[d, "\mathrm{List}\, f"] \\
\mathrm{List}\, B
\end{tikzcd}
\qquad
\begin{tikzcd}
{[a_1, a_2, ...a_n]} \arrow[d, mapsto, "\mathrm{List}\, f"] \\
{[f(a_1), f(a_2), ..., f(a_n)]}
\end{tikzcd}.$$

From a categorical perspective, the functor List can be presented as follows:

$$\begin{tikzcd}
\mathcal{S}et \arrow[r, "\Delta"] \arrow[d, "\mathrm{List}"'] & \mathcal{S}et^{\mathbb{N}} \arrow[d, "\prod_{n\in\mathbb{N}} \times^n"] \\
\mathcal{S}et & \mathcal{S}et^{\mathbb{N}} \arrow[l, "\coprod_{n\in\mathbb{N}}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, mapsto] \arrow[d, mapsto, "\mathrm{List}(A)"'] & A^{\mathbb{N}} \arrow[d, mapsto] \\
\coprod_{n\in\mathbb{N}} A^n & (A^n)_{n\in\mathbb{N}} \arrow[l, mapsto]
\end{tikzcd}.$$

//TODO: Specify the Δ framework

This presentation gives rise to the following API:

- The collection of constructors

$$[\ ]_{n,A} := A^n \xrightarrow{i_n} \mathrm{List}\, A := \lambda a_1 \dots a_n.[a_1, \dots, a_n],$$

  in particular:

  - The single-element list constructor

$$\eta_A : A \to \mathrm{List}\, A := \lambda a.[a],$$

  - The empty list constructor

$$[\ ]_0 : 1 \to \mathrm{List}\, A := \lambda a.[];$$

- Length of a list via the power of A in the coproduct:

$$\mathrm{len}_A : \mathrm{List}\, A \to \mathbf{N};$$

– List concatenation via the natural isomorphism $A^n \times A^m \cong A^{n+m}$:

$$\begin{tikzcd} \mathrm{id} \times \mathrm{id} \arrow[r, "i \times i"] \arrow[d, "\simeq"'] & \mathrm{List} \times \mathrm{List} \arrow[d, "+"] \\ \mathrm{id} \arrow[r, "i"'] & \mathrm{List} \end{tikzcd} \qquad \begin{tikzcd} A^m \times A^n \arrow[r, "i_m \times i_n"] \arrow[d, "\cong"'] & \mathrm{List}\, A \times \mathrm{List}\, A \arrow[d, "+_A"] \\ A^{m+n} \arrow[r, "i_{m+n}"'] & \mathrm{List}\, A \end{tikzcd},$$

$$+_A : \mathrm{List}\, A \times \mathrm{List}\, A \to \mathrm{List}\, A := \lambda lm.[l_0, ...l_{\mathrm{len}(l)-1}, m_0, ..., m_{\mathrm{len}(m)-1}],$$
$$+_A : (\mathrm{List}\, A)^n \to \mathrm{List}\, A := \lambda l_0...l_n.l_0 + ... + l_n;$$

– List destructors:

$$\mu_A : \mathrm{List}(\mathrm{List}\, A) \to \mathrm{List}(A) := \lambda l. \sum_{i=0}^{\mathrm{len}(l)-1} l_i,$$
$$\mu_A : \mathrm{List}^n\, A \to \mathrm{List}\, A := \mu_A^{\circ n}.$$

First of all, this API provides an internal monoid structure on lists: for a fixed set A, we have a monoid object

$$(\mathrm{List}\, A, [], +_A)$$

in $\mathcal{S}et$. The corresponding diagrams are straightforward in this case.

Second, it provides a monad structure on List:

$$(\mathrm{List}, \eta : \mathrm{id} \to \mathrm{List}, \mu : \mathrm{List}^2 \to \mathrm{List}).$$

The corresponding diagrams:

$$\begin{tikzcd} \mathrm{List}\, A \arrow[r, "\eta_{\mathrm{List}\, A}"] \arrow[dr, "\mathrm{id}"'] & \mathrm{List}(\mathrm{List}\, A) \arrow[d, "\mu_A"] \\ & \mathrm{List}\, A \end{tikzcd} \qquad \begin{tikzcd} l \arrow[r, mapsto] \arrow[dr, mapsto] & {[l]} \arrow[d, mapsto] \\ & l = [l]_0 \end{tikzcd},$$

$$\begin{tikzcd} \mathrm{List}(\mathrm{List}\, A) \arrow[d, "\mu_A"'] & \mathrm{List}\, A \arrow[l, "\mathrm{List}\, \eta_A"'] \arrow[dl, "\mathrm{id}"] \\ \mathrm{List}\, A & \end{tikzcd} \qquad \begin{tikzcd} {[[l_0], ..., [l_{\mathrm{len}(l)-1}]]} \arrow[d, mapsto] & l \arrow[l, mapsto] \arrow[dl, mapsto] \\ l = \sum_{i=0}^{\mathrm{len}(l)-1} [l_i] & \end{tikzcd},$$

$$\begin{tikzcd} \mathrm{List}^3\, A \arrow[r, "\mathrm{List}\, \mu_A"] \arrow[d, "\mu_{\mathrm{List}\, A}"'] & \mathrm{List}^2\, A \arrow[d, "\mu_A"] \\ \mathrm{List}^2\, A \arrow[r, "\mu_A"'] & \mathrm{List}\, A \end{tikzcd} \qquad \begin{tikzcd} l \arrow[r, mapsto] \arrow[d, mapsto] & {[\sum_{j=0}^{\mathrm{len}(l_0)-1} (l_0)_j, ..., \sum_{j=0}^{\mathrm{len}(l_{\mathrm{len}(l)-1}-1)} (l_{\mathrm{len}(l)-1})_j]} \arrow[d, mapsto] \\ \sum_{i=0}^{\mathrm{len}(l)-1} l_i \arrow[r, mapsto] & \sum_{i,j} (l_i)_j \end{tikzcd}.$$

3 Comonads