NOTES ON THE SEQUENCE MAP FOR MONADS

IORDAN GANEV

Version 1.2

Contents

1. Introduction

These notes give a formulation of the categorical semantics behind the `sequence` map in Haskell. While our focus is on the mathematical foundations, we also have practical goals of understanding how best to use the `Monad` type class. We assume familiarity with the basics of category theory, and exposure to Haskell. Recommended resources include [oH, Rie17, Mil19].

1.1. Monads in Haskell.

Haskell is a strongly typed language, and contains various ways to produce new types based on old types; these are known as *type constructors*. Roughly speaking, a monad `m` is a type constructor with desirable properties with respect to composition. The monad properties allow for a consistent and associative way to "compose" a function of type `a -> m b` with one of type `b -> m c` to produce one of type `a -> m c`:

```
(a -> m b) -> (b -> m c) -> a -> m c
```

In many cases, it is helpful to regard values of type `m a` as operations that can be executed to produce a value of type `a`. So the composition rule can be interpreted giving a consistent way to combine two executions into one. We discuss monads in Haskell in more detail in Section 4. Examples include:

- The maybe type constructor `Maybe a = Nothing | Just a`.
- The list type constructor `[a] = Nil | Cons a [a]`.

- The state type constructor `State s a = State (s -> (a,s))`, where `s` is the state and `a` is the argument of the monad.
- The representable type constructor `Repr e a = Repr (e -> a)` wher `e` is considered fixed and `a` is the argument of the monad.

1.2. **Monads in category theory.** In category theory, a *monad* on a category $\mathcal{C}$ is an endofunctor $T : \mathcal{C} \to \mathcal{C}$ with properties that allow for a consistent and associative composition rule:

$$\mathcal{C}(X, TY) \times \mathcal{C}(Y, TZ) \to \mathcal{C}(X, TZ)$$

where we write $\mathcal{C}(-,-)$ for the set of morphisms from one object to another. This composition rule is part of the definition of the *Kleisli category* associated to the monad T, which we discuss further in Section 2. Moreover, the composition rule is equivalent to requiring that T be a monoid in the category of endofunctors (with the operation of composition), which is the usual method of introducing monads in category theory. In Section 5, we discuss the following examples of monads on the category of sets:

(1) The maybe monad $TX = 1 + X$.
(2) The list monad $TX = [X] = \sum_{n \in \mathbb{N}} X^n$.
(3) The state monad $TX = \mathrm{Hom}(S, X \times S)$ for a fixed set S.
(4) The free module monad $TX = \mathrm{Hom}_{\mathrm{f}}(X, R)$ where R is a semiring (such as the natural numbers $\mathbb{N}$ or the real numbers $\mathbb{R}$), and the subscript "f" denotes finitely supported functions.
(5) The representable monad $TX = \mathrm{Hom}(E, X)$ for a fixed set E.

1.3. **The sequence map.** In Haskell, the sequence map has type `[ma]->m[a]` and can be thought of as converting a list of monadic operations, each producing a value of type `a`, to a single monadic operation that produces a list of values of type `a`, that is, it produces a value of type `[a]`. We give a categorical formulation of this procedure in Section 3, its implementation in Haskell in Section 4, and examples in Section 5. A proposed generalization of the sequence map appears in Appendix B.

1.4. **Notation.** For a category $\mathcal{C}$, we write $\mathcal{C}(X, Y)$ for the set of morphisms from the object X to the object Y. We use T to denote a monad on category $\mathcal{C}$, and write the multiplication and unit natural transformations as $\mu : T^2 \to T$ and $\epsilon : 1 \to T$, respectively. These satisfy the associativity and unit axioms:

$$\mu_X \circ \mu_{TX} = \mu_X \circ T\mu_X \qquad \mu_X \circ \epsilon_{TX} = \mathrm{id}_{TX} = \mu_X \circ T\epsilon_X$$

In the context of Haskell, we will denote a monad as `m`.

When working in the category of sets, we write $X + Y$ for the coproduct (disjoint union) of two sets X and Y, which are referred to as the cofactors of the coporduct. Given maps $f : X \to Z$ and $g : Y \to Z$, we write $\langle f, g \rangle$ for the induced map $X + Y \to Z$. We also write $\sum_{i \in I} X_i$ for the coproduct of an indexed family of sets.

Our use of the symbol 1 is overloaded; its meaning will be clear from context to be either the identity functor, the identity map, the one-point set, the single element of the one-point set, or the number one.

2. The Kleisli category

Let T be a monad on a category $\mathcal{C}$. The Kleisli category $\mathcal{C}_T$ is defined as having the same objects as $\mathcal{C}$, with morphisms given by:

$$\mathcal{C}_T(X, Y) = \mathcal{C}(X, TY)$$

The identity in $\mathcal{C}_T(X, X)$ is given by the unit $\epsilon_X : X \to TX$, and the composition of $f_1 : X \to TY$ and $f_2 : Y \to TZ$ is given by:

$$\begin{aligned} f_2 \circ_T f_1 &:= \mu_Z \circ Tf_2 \circ f_1 \\ &= (X \xrightarrow{f_1} TY \xrightarrow{Tf_2} T^2Z \xrightarrow{\mu_Z} TZ) \end{aligned}$$

The associativity of composition, as well as the identity properties, follow from (and are in fact equivalent to) the associativity and unit axioms of the monad T. (See Exercise 3.) There are functors:

$$\mathcal{C} \underset{U_T}{\overset{F_T}{\rightleftarrows}} \mathcal{C}_T$$

defined as follows:

$$F_T X = X \qquad\qquad U_T X = TX$$

$$F_T(X \xrightarrow{f} Y) = X \xrightarrow{f} Y \xrightarrow{\epsilon_Y} TY \qquad\qquad U_T(X \xrightarrow{g} TY) = TX \xrightarrow{Tg} T^2Y \xrightarrow{\mu_Y} TY$$

In other words, F_T is the identity on objects and post-composes with the unit on morphisms: $F_T f = \epsilon_Y \circ f$. Meanwhile, U_T applies T to objects and, on morphisms, applies T and then multiplies: $U_T g = \mu_Y \circ Tg$.

Lemma 2.1. *We have:*

(1) *The functors above form an adjoint pair (F_T, U_T):*

$$\mathcal{C}_T(F_T X, Y) = \mathcal{C}_T(X, Y) = \mathcal{C}(X, TY) = \mathcal{C}(X, U_T Y).$$

(2) *The monad of the adjunction is $T = U_T \circ F_T$. In particular,*

$$Tf = U_T(\epsilon_Y \circ f), \qquad \mu_X = U_T(\mathrm{id}_{TX}), \qquad \epsilon_X = F_T(\mathrm{id}_X).$$

(3) *The functor U_T applied to morphisms defines a natural transformation*

$$\mathcal{C}(-, T-) \to \mathcal{C}(T-, T-)$$

of functors $\mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$.

(4) *For any $g = g_1 : X \to TY$ and $g_2 : Y \to TZ$, we have:*

$$(U_T g) \circ \epsilon_X = g \qquad U_T(\epsilon_X) = \mathrm{id}_{TX} \qquad U_T(U_T(g_2) \circ g_1) = U_T(g_2) \circ U_T(g_1)$$

Sketch of proof. The first claim follows from definitions. The second also follows from definitions, together with the unit axiom to show that $U_T(\epsilon_Y \circ f) = \mu_Y \circ T\epsilon_Y \circ Tf = Tf$. The third claim is a consequence of the facts that (1) U_T is a functor, (2) U_T applies

T to objects, and (3) $\mathcal{C}_T(X,Y) = \mathcal{C}(X,TY)$. The identities of the final claim are easily verified using (1) the definition that $U_T g = \mu_Y \circ Tg$, (2) the fact that μ and ϵ are natural transformations, and (3) the unit axiom for T. □

Remark 2.2. There is a precise sense in which the Kleisli category, equipped with the adjunction (F_T, U_T) is initial among all adjunctions giving rise to the monad T. The category of T-algebras (also known as the Eilenberg–Moore category) is final.

3. The sequence map

In this section, we specialize to the category Set of sets, and write $\mathrm{Hom}(X,Y) = \mathsf{Set}(X,Y)$ for the set of morphisms from a set X to a set Y. Recall the product-hom adjunction:

$$\mathrm{Hom}(X, \mathrm{Hom}(Y,Z)) \simeq \mathrm{Hom}(X \times Y, Z) \simeq \mathrm{Hom}(Y, \mathrm{Hom}(X,Z))$$

for any sets X, Y, and Z.

3.1. **The list endofunctor.** Let $[X] = \mathsf{List}(X) = \sum_{n=0}^{\infty} X^n$ be the set of all lists with entries in the set X; this defines an endofunctor[1] $[-] : \mathsf{Set} \to \mathsf{Set}$, where a map $f : X \to Y$ is sent to the map $[x_1, \ldots, x_n] \mapsto [f(x_1), \ldots, f(x_n)]$.

Given a list in $[X]$ and element in X, we can form a new list by attaching the element to the front of the list. This is traditionally known as the cons map:

$$\mathsf{cons}_X : X \times [X] \to [X]; \qquad (x, [x_1, \ldots, x_n]) \mapsto [x, x_1, \ldots, x_n]$$

The image of cons_X comprises all non-empty lists. Following Haskell notation, we abbreviate $\mathsf{cons}(x, \mathbf{x})$ by $(x : \mathbf{x})$. Including the empty list via the map $\mathsf{nil}_X : 1 \to [X]$ gives an isomorphism:

$$\langle \mathsf{nil}_X, \mathsf{cons}_X \rangle : 1 + X \times [X] \xrightarrow{\sim} [X]$$

where we apply nil_X to the first cofactor and cons_X to the second.

3.2. **The map $T^{(2)}$.** Let T be a monad on Set, and define a natural transformation

$$T^{(2)} : \mathrm{Hom}(- \times -, -) \to \mathrm{Hom}((T-) \times (T-), T-)$$

of functors $\mathsf{Set}^{\mathrm{op}} \times \mathsf{Set}^{\mathrm{op}} \times \mathsf{Set} \to \mathsf{Set}$ as:

$$\begin{aligned} T^{(2)} = T^{(2)}_{X,Y,Z} : \mathrm{Hom}(X \times Y, Z) &\xrightarrow{\sim} \mathrm{Hom}(Y, \mathrm{Hom}(X,Z)) \\ &\xrightarrow{T \circ -} \mathrm{Hom}(Y, \mathrm{Hom}(TX, TZ)) \\ &\xrightarrow{\sim} \mathrm{Hom}(TX, \mathrm{Hom}(Y, TZ)) \\ &\xrightarrow{U_T \circ -} \mathrm{Hom}(TX, \mathrm{Hom}(TY, TZ)) \\ &\xrightarrow{\sim} \mathrm{Hom}(TX \times TY, TZ) \end{aligned}$$

[1] In fact, List is a monad, but its monad structure is not relevant in this section.

The first, third, and fifth maps use the adjunction $(\times, \mathrm{Hom})$; the second applies the functor T to morphisms; the fourth uses U_T applied to $\mathsf{Set}_T(Y, Z) = \mathrm{Hom}(Y, TZ)$. More explicitly, we have:

$$T^{(2)}(f)(p,q) = U_T(x \mapsto Tf(x,-)(q))(p).$$

for $f : X \times Y \to Z$ and $(p,q) \in TX \times TY$. Since $Tf(x,-) = U_T(\epsilon_Z \circ f(x,-))$, we can equivalently write:

$$T^{(2)}(f)(p,q) = U_T(x \mapsto U_T(\epsilon_Z \circ f(x,-))(q))(p).$$

3.3. **The sequence map.** We now define the sequence natural transformation

$$\mathsf{seq} : [T-] \to T[-]$$

of endofunctors. Given a set X, we have the isomorphism

$$\langle \mathsf{nil}_{TX}, \mathsf{cons}_{TX} \rangle : 1 + TX \times [TX] \xrightarrow{\sim} [TX],$$

and so it is enough to define $\mathsf{seq}_X : [TX] \to T[X]$ on the empty list and on a list of the form $(p : \mathbf{p})$ for $p \in TX$ and $\mathbf{p} \in [TX]$. We do this as follows:

$$\begin{aligned} \mathsf{seq} : [TX] &\to T[X] \\ [\,] &\mapsto \epsilon_{[X]}([\,]) \\ (p : \mathbf{p}) &\mapsto T^{(2)}_{X,[X]}(\mathsf{cons})(p, \mathsf{seq}(\mathbf{p})) \end{aligned}$$

where we note that $T^{(2)}_{X,[X]}(\mathsf{cons})$ is a map $TX \times T[X] \to T[X]$. Equivalently, we have the following commutative diagram:

$$\begin{array}{ccc} 1 + TX \times [TX] & \xrightarrow{\langle\, 1 \,,\, 1 \times \mathsf{seq}_X \,\rangle} & 1 + TX \times T[X] \\ {\scriptstyle \langle\, \mathsf{nil}_{TX} \,,\, \mathsf{cons}_{TX} \,\rangle}\downarrow & & \downarrow{\scriptstyle \langle\, \epsilon_{[X]} \circ \mathsf{nil}_X \,,\, T^{(2)}_{X,[X]}(\mathsf{cons}_X) \,\rangle} \\ [TX] & \xrightarrow{\mathsf{seq}_X} & T[X] \end{array}$$

Remark 3.1. In appendix B, we propose a generalization of the sequence map to a natural transformation $E \circ T \to T \circ E$ to a class of functors E beyond the list endofunctor.

4. Monads in Haskell

4.1. **Type constructors and functors.** Haskell is a strongly-typed language, which means that every Haskell value and expression has a type. Haskell also features type constructors that take a type and produce a new type. Examples include the `Maybe` constructor, the `List` constructor[2], and the binary tree constructor `BTree`:

```
data Maybe a = Nothing | Just a
data List a = Nil | Cons a (List a)
data BTree a = Empty | Node a (BTree a) (BTree a)
```

A type constructor may also take in multiple types to produce the new type, as in:

[2]The list type with entries of type `a` is usually denoted `[a]`.

```
data Either a b = Left a | Right b
data ListEither a b = Nil | Cons1 a (ListEither a b) | Cons2 b (
   ListEither a b)
```

In these notes, we focus on single-variable type constructors. A type constructor is a `Functor` in Haskell if it is equipped with a way of producing a new function for every function between the type variables. The minimal complete definition is:

```
class Functor f where
  fmap :: (a -> b) -> f a -> f b
```

To connect with category theory, the relevant category is the category of Haskell types. We cannot do justice to the deep theory behind this category; to avoid a rabbit hole of technicalities, not much is lost if one thinks of the category of Haskell types as simply the category of sets. A `Functor` in the sense of Haskell is an endofunctor of this category if it satisfies the identity and composition axioms:

```
fmap id = id                          -- identity axiom
fmap (f . g)  ==  fmap f . fmap g     -- composition axiom
```

Functors defined in Haskell are expected to respect identities and composition, though there is no way to enforce this in Haskell. Conversely, an endofunctor F on Set defines a natural transformation $\mathrm{Hom}(-,-) \to \mathrm{Hom}(F-,F-)$ of functors $\mathsf{Set}^{\mathrm{op}} \times \mathsf{Set} \to \mathsf{Set}$; this corresponds to `fmap`. The `Maybe`, `List`, and `BTree` constructors are "naturally" functors.

4.2. **Monads.** A type constructor `m` is considered a `Monad` in Haskell if it is equipped with (1) a way of extending any function `a -> m b` to a function `m a -> m b`, and (2) a map from any type `a` to the output type `m a`. The minimal complete definition is:

```
class Monad m where
  (>>=)  :: m a -> (  a -> m b) -> m b
  return ::   a                 -> m a
```

A monad in Haskell is expected to satisfy the following identities (which cannot actually be enforced in Haskell):

```
-- x::a     z::m a     f::a -> m b     g::b -> m c
return x >>= f                  =  f x              -- left identity
z        >>= return             =  z                -- right identity
z        >>= (\x -> f x >>= h)  =  (z >>= f) >>= h  -- associativity
```

Moreover, a `Monad` in Haskell is in particular a `Functor`:

```
liftM :: Monad m => (a -> b) -> (m a -> m b)
liftM f z = z >>= (return . f)
```

The identity and composition laws for the `liftM` follow from the identity and associativity laws for the monad `m`. Table 1 gives a dictionary between constructions and identities in Haskell and concepts in category theory.

Haskell	Category Theory
Monad `m`	Monad $T : \mathrm{Set} \to \mathrm{Set}$
`f::a->m b`	$f : X \to TY$
`z >>= f`	$U_T(f)(z)$
`return::a->m a`	$\epsilon : 1 \to T$
`prod::m(ma)->ma, (p -> p >>= id)`	$\mu_X : T^2X \to TX$
`return x >>= f = f x`	$(U_T f) \circ \epsilon_X = f$
`z >>= return = z`	$U_T(\epsilon_X) = \mathrm{id}_{TX}$
`z>>=(\x->f x>>=h) = (z>>=f)>>=h`	$U_T(U_T(h) \circ f) = U_T(h) \circ U_T(f)$
`liftM f z = z >>= (return . f)`	$Tf = U_T(\epsilon_Y \circ f)$

FIGURE 1. Dictionary between Haskell and category theory.

It is easy to see from Lemma 2.1 that a monad in the sense of category theory defines a `Monad` in the sense of Haskell. In Appendix A, we give a proof of the converse: a `Monad` in Haskell gives rise to a ordinary category theory monad as long as the identity and associativity axioms hold.

We give an implementation of the composition rule mentioned in Section 1:

```
compose :: (a -> m b) -> (b -> m c) -> a -> m c
compose g1 g2 x = (g1 x) >>= g2
```

Remark 4.1. Haskell also has an operator `(>>):ma->mb->mb` for a monad `m`. This can be recovered from `(>>=)` by using a constant function in the second factor:

```
z>>w  = z>>= (\x -> w)
```

4.3. **`do` blocks.** Haskell features `do` blocks as a way to abbreviate compositions in the Kleisli category and the functor U_T. To illustrate the motivation, let `z:ma` and `w:mb`. We will often want to perform computations like:

```
z >>= [\x -> w >>= [\y -> u x y]]
```

where `z::ma`, `w::mb`, and `u` is a function of type `a->b->mc`. Walking through this: given `x::a`, we have the function `b->mc` taking `y` to `u x y`. This function feeds in as the second input of the operator `>>=` with `w::mb` being the first input. The operator returns an output of type `mc`. Recalling that `x::a` was fixed, we see that we obtain a function `a->mc`. This function in turn feeds as the second input of the operator `>>=` with `z::ma` being the first input. Hence, we the final output is of type `mc`.

Using `do` blocks, this simplifies to:

```
do
  x <- z
  y <- w
  u x y
```

In general, a code block of the form

```
do
  x <- z
  ...
```

is equivalent to:

```
z >>= [\x -> do ...]
```

Some notes:

- The final line of a `do` block must be an expression of type `m a` for some type `a`. One can think of the type of the entire `do` block as having the type of the last line.
- A non-final line may be of the form `v <- expr`, where `expr` has type `m a` for some type `a`. The variable `v` is a dummy variable of type `a` used to define a function with input type `a`.
- Alternatively, a non-final line may be consist of just an `expr` of type `m a` for some type `a`. This line can be replaced by `v <- expr` with no change in the functionality of the code block. (This corresponds to defining a constant function and can be avoided with the operator `>>`, see remark 4.1.)
- The remaining lines are `let` statements, which are less relevant for categorical considerations.

Executing a `do` statement amounts to performing a composition of morphisms in the Kleisli category. Conceptually (though not necessarily practically), one can think of going from the last line to the first.

4.4. **The sequence map in Haskell.** The idea of the sequence function is to capture the following function concept in (pseudo-)Haskell:

```
sequence :: Monad m => [m a] -> m [a]
sequence [] = return []
sequence [x1,...,xn] = do   x1 <- p1
                            x2 <- p2
                            ...
                            xn <- pn
                            return [x1,x2,...,xn]
```

Obviously, the above doesn't compile; an actual implementation of the sequence map is:

```
sequence :: Monad m => [m a] -> m [a]
sequence [] = return []
sequence (p:ps) = p >>= \x -> (sequence ps) >>= \ys -> return (x:ys)
```

4.5. **The sequence map as a fold.** To connect with Section 3, let us implement the map $T^{(2)}$, as well as nil and cons:

```
T2 :: (a -> b -> c) -> m a -> m b -> m c
T2 f z u = z >>= (\x -> u >>= return . f x)

nil = [] :: [a]

cons :: a -> [a] -> [a]
cons x ys = x:ys
```

Taking `b` = `c` = `[a]`, this leads us to implement the sequence function as a fold:

```
sequence :: Monad m => [m a] -> m [a]
sequence = foldr (T2 cons) (return nil)
```

There is a way in which nil can be regarded as $T^{(0)}$ (see Appendix B), so that the fold becomes `foldr (T2 cons) (T0 nil)`.

4.6. **`MapM`.** Haskell also has the function `MapM`, which can be defined as:

```
mapM :: Monad m => (a -> m b) -> [a] -> m [b]
mapM f = sequence . map f
```

We note in passing, that, for performance reasons, Haskell implements `mapM` directly and then defines `sequence = mapM id`.

5. Examples

Here are examples of monads on the category of sets. For each, we comment on:

- the monad structure maps μ and ϵ,
- the Kleisli composition $TX \times \mathrm{Hom}(X, TY) \to TY$ taking (z, g) to $U_T(g)(z)$,
- the map $T^{(2)} : \mathrm{Hom}(X \times Y, Z) \to \mathrm{Hom}(TX \times TY, TZ)$, and
- the sequence map $\mathrm{seq}_X : [TX] \to T[X]$.

5.1. **The maybe monad.** Consider the functor $TX = 1 + X$, which acts on morphisms by extending a function $f : X \to Y$ to the function $1 + X \to 1 + Y$ sending 1 to 1. This defines a monad with $\epsilon : X \to 1 + X$ including into the second cofactor, and $\mu : T^2X = 1 + 1 + X \to 1 + X$ collapsing both factors of 1 into the single one. The Kleisli composition map:

$$TX \times \mathrm{Hom}(X, TY) = (X + 1) \times \mathrm{Hom}(X, 1 + Y) \to TY = 1 + Y$$

extends a function $X \to 1 + Y$ by sending 1 to 1. The map:

$$T^{(2)} : \mathrm{Hom}(X \times Y, Z) \to \mathrm{Hom}(TX \times TY, TZ) = \mathrm{Hom}(1 + X + Y + X \times Y, 1 + Z)$$

extends a function $X \times Y \to Z$ by sending the cofactors 1, X, and Y to 1. More generally, to describe the sequence map, let $[k_1, \ldots, k_n]$ be a list with entries in $1 + X$, so that each k_i is either equal to 1 or belongs to X. Then:

$$\mathsf{seq}([k_1, \ldots, k_n]) = \begin{cases} [x_1, \ldots, x_n] \in [X] \subseteq 1 + [X] & \text{if } k_i = x_i \in X \text{ for all } i \\ 1 \in 1 + [X] & \text{otherwise} \end{cases}$$

In other words, if any of the k_i's is equal to one, then the output of the sequence map is 1, and otherwise the sequence map is the identity.

5.2. **The list monad.** The list functor $TX = [X] = \mathsf{List}(X) = \sum_{n=0}^{\infty} X^n$ defines a monad with $\epsilon : X \to [X]$ including X as singleton lists, and $\mu : T^2X \to TX$ concatenating a list of lists into a single list. The Kleisli composition:

$$[X] \times \mathrm{Hom}(X, [Y]) \to [Y]$$

takes a list $[x_1, \ldots, x_n]$ and a function $f : X \to [Y]$ to the concatenation of the lists $f(x_1), ..., f(x_n)$. The map

$$T^{(2)} : \mathrm{Hom}(X \times Y, Z) \to \mathrm{Hom}([X] \times [Y], [Z])$$

takes $f : X \times Y \to Z$ to the function taking the pair of lists $(\mathbf{x}, \mathbf{y})$ to the list consisting of all f evaluated on all combinations of pairs of elements formed by taking one element of $\mathbf{x} = [x_1, \ldots, x_n]$ and one element of $\mathbf{y} = [y_1, \ldots, y_m]$:

$$[f(x_i, y_j) \ \mid \ i = 1, \ldots, n; j = 1, \ldots, m]$$

In particular, the length of the output list is the product of the lengths of the input lists. More generally, the sequence map takes a list of lists $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$ to the list of all lists of length n whose i-th entry is an entry of $\mathbf{x}_i$.

$$\mathsf{seq}([\mathbf{x}_1, \ldots, \mathbf{x}_n]) = [[\mathbf{x}_{1,i_1}, \mathbf{x}_{2,i_2}, \ldots, \mathbf{x}_{n,i_n}] \ \mid \ i_1 = 1, \ldots, \mathsf{len}(\mathbf{x}_i), \ldots, i_n = 1, \ldots, \mathsf{len}(\mathbf{x}_n)]$$

The number of such lists is the product of the lengths of the $\mathbf{x}_i$. If any of the input lists is empty, then the output list is empty.

We remark that the maybe monad is in some sense a special case of the list monad as there is an inclusion $1 + X \hookrightarrow [X]$ as the lists of length at most one.

5.3. **The state monad.** Fix a set S, the "state". The functor

$$TX = \mathrm{Hom}(S, X \times S) \simeq \mathrm{Hom}(S, X) \times \mathrm{End}(S)$$

can be thought of all ways to update the state (the $\mathrm{End}(S)$ factor) and producing a "value" in the set X. It takes $f : X \to Y$ to the map that post-composes with $f \times 1$.

This functor defines a monad with $\epsilon : X \to \mathrm{Hom}(S, X \times S)$ taking x to the function $s \mapsto (x, s)$ for all s. The product

$$\mu : X \to \mathrm{Hom}(S, \mathrm{Hom}(S, X \times S) \times S) \to \mathrm{Hom}(S, X \times S)$$

is post-composition with the evaluation map $\mathrm{Hom}(S, X \times S) \times S \to X \times S$. The Kleisli composition:

$$\mathrm{Hom}(S, X \times S) \times \mathrm{Hom}(X, \mathrm{Hom}(S, Y \times S)) \to \mathrm{Hom}(S, Y \times S)$$

is equivalent, via the $(\times, \mathrm{Hom})$ adjunction, to a map

$$\mathrm{Hom}(S, X \times S) \times \mathrm{Hom}(X \times S, Y \times S) \to \mathrm{Hom}(S, Y \times S)$$

and is given by composition. The map

$$T^{(2)} : \mathrm{Hom}(X \times Y, Z) \to \mathrm{Hom}(\mathrm{Hom}(S, X \times S) \times \mathrm{Hom}(S, Y \times S), \mathrm{Hom}(S, Z \times S))$$

takes $f : X \times Y \to Z$ to the function mapping the pair $(\phi : S \to X \times S\ ,\ \psi : S \to Y \times S)$ to the composition:

$$S \xrightarrow{\phi} X \times S \xrightarrow{1 \times \psi} X \times Y \times S \xrightarrow{f \times 1} Z \times S$$

The sequence map $\mathsf{seq}_X : [\mathrm{Hom}(S, X \times S)] \to \mathrm{Hom}(S, [X] \times S)$ takes a list of functions $[\phi_1, \ldots, \phi_n]$ to the function

$$s \mapsto ([x_1, \ldots, x_n], s_n)$$

where the $x_i \in X$ and $s_i \in S$ are defined recursively via:

$$s_0 = s, \qquad \text{and} \qquad (x_i, s_i) = \phi_i(s_{i-1}) \quad \text{for } i = 1, \ldots, n.$$

In other words, the x_i are the values produced while successively updating the state, and s_n is the final state. Equivalently, $\mathsf{seq}_X([]) = [s \mapsto ([], s)]$ and $\mathsf{seq}_X(\phi : \overline{\phi}) = [s \mapsto (x : \mathbf{x}, r)]$ where $\mathsf{seq}_X(\overline{\phi})(s) = (\mathbf{x}, t)$ and $\phi(t) = (x, r)$.

5.4. **The free semigroup module monad.** Recall that a semiring is defined in the same way as a ring, except without the assumption of additive inverses. Let R be a semiring and consider the functor

$$TX = \mathrm{Hom}_{\mathrm{f}}(X, R),$$

where $\mathrm{Hom}_{\mathrm{f}}$ denotes the set of functions with finite support. This is a covariant functor via pushforward of functions:

$$T(X \xrightarrow{f} Y) : \mathrm{Hom}_{\mathrm{f}}(X, R) \to \mathrm{Hom}_{\mathrm{f}}(Y, R), \qquad \phi \mapsto f_*\phi = \left[y \mapsto \sum_{x \in f^{-1}(y)} \phi(x) \right]$$

Relevant instances of this functor are:

- The natural numbers $\mathbb{N}$. In this case, $TX = \mathsf{Bag}(X)$ is the Bag endofunctor. A bag of elements of X can also be thought of a multi-set; every element has some multiplicity. There is a forgetful natural transformation of monads $\mathsf{List} \to \mathsf{Bag}$.
- The real numbers $\mathbb{R}$ (or any field). In this case, TX is the vector space with basis given by the elements of X.
- The Boolean set $\{0, 1\}$ with OR as addition and AND as multiplication. In this case, $TX = \mathcal{P}_{\mathrm{f}}(X)$ is the set of finite subsets of X.

Observe that TX is the free R-module on the set X, and Tf is a linear map for any $f : X \to Y$. If X is finite, then $TX \simeq R^{|X|}$ is itself a semiring with coordinate-wise operations. If X is not finite, then TX is a semiring without multiplicative unit.

We now discuss the monad structure. The unit $\epsilon_X : X \to \mathrm{Hom}_f(X, R)$ picks out the delta functions, so that $\epsilon_X(x)(x') = 1$ if $x = x'$ and zero otherwise. In other symbols, $\epsilon_X(x) = \delta_x$. The monad product is given by:

$$\begin{aligned} \mu_X : \mathrm{Hom_f}(\mathrm{Hom_f}(X, R), R) &\to \mathrm{Hom_f}(X, R) \\ \Phi &\mapsto \left[x \mapsto \sum_{\phi \in \mathrm{Hom_f}(X,R)} \Phi(\phi)\phi(x) \right] \end{aligned}$$

The sum is well-defined due to the finite support condition. The Kleisli composition is given by:

$$\begin{aligned} \mathrm{Hom_f}(X, R) \times \mathrm{Hom}(X, \mathrm{Hom_f}(Y, R)) &\to \mathrm{Hom_f}(Y, R) \\ (\phi, f) &\mapsto \left[y \mapsto \sum_{x \in X} \phi(x) f(x)(y) \right] \end{aligned}$$

Note that (δ_x, f) is sent to $f(x)$. The Kleisli composition can be thought of as a convolution; let $\pi_1 : X \times Y \to X$ and $\pi_2 : X \times Y \to Y$ be the projection maps. Then (ϕ, f) goes to $(\pi_2)_*(\pi_1^* \cdot f)$, where π_1^* is the pullback, $(\pi_2)_*$ is the pushforward, we regard f as a function $X \times Y \to R$, and $\cdot$ denotes point-wise multiplication.

Moving on, we have:

$$\begin{aligned} T^{(2)} : \mathrm{Hom}(X \times Y, Z) &\to \mathrm{Hom}(\mathrm{Hom_f}(X, R) \times \mathrm{Hom_f}(Y, R), \mathrm{Hom_f}(Z, R)) \\ f &\mapsto \left[(\phi, \psi) \mapsto \left[z \mapsto \sum_{(x,y) \in f^{-1}(z)} \phi(x)\psi(y) \right] \right] \end{aligned}$$

Another interpretation: (ϕ, ψ) defines a function $\phi \times \psi : X \times Y \to R \times R$, which we can post-compose with the multiplication map $\mathrm{mult}_R : R \times R$. Then $T^{(2)} f$ sends (ϕ, ψ) to the pushforward $f_*(\mathrm{mult}_R \circ (\phi \times \psi))$. We make two final remarks about $T^{(2)} f$: (1) it is not linear, but bilinear, (2) it sends (δ_x, δ_y) to $\delta_{f(x,y)}$.

Finally, we the sequence map is given by:

$$\begin{aligned} \mathsf{seq}_X : [\mathrm{Hom}(X, R)] &\to \mathrm{Hom}([X], R) \\ [\phi_1, \ldots, \phi_n] &\mapsto \left[\mathbf{x} \mapsto \begin{cases} \prod_{i=1}^n \phi_i(x_i) & \text{if } \mathrm{len}(\mathbf{x}) = n \\ 0 & \text{otherwise} \end{cases} \right] \end{aligned}$$

In particular, the sequence map factors through a map $\mathrm{Hom}(X, R)^n \to \mathrm{Hom}(X^n, R)$, and sends a list of delta functions to the delta function on the corresponding list: $[\delta_{x_1}, \ldots, \delta_{x_n}] \mapsto \delta_{[x_1,\ldots,x_n]}$. For $n = 0$, we have that $\mathsf{seq}_X([]) = \delta_{[]}$ is the delta function at the empty set. We revisit the relevant instances of this functor:

- For $R = \mathbb{N}$, the sequence map corresponds to the well-known natural transformation $\mathsf{List} \circ \mathsf{Bag} \to \mathsf{Bag} \circ \mathsf{List}$. Given a list of n bags, the multiplicity of a list in the output bag of lists is the number of ways the list can be realized via taking the first element from the first bag, the second from the second bag, etc. So only lists of length n can have non-zero multiplicity.

- If R is a field, then the sequence map restricts to the natural map from the direct sum of n copies of a vector space to the tensor product of n copies of the same vector space: $V^{\oplus n} \to V^{\otimes n}$ taking $(v_1, \ldots, v_n)$ to $v_1 \otimes \cdots \otimes v_n$.

- Finally, for the finite power set monad, the sequence map takes a list of subsets $[S_1, \ldots, S_n]$ of X to the subset of $[X]$ consisting of all lists $[x_1, \ldots, x_n]$ where x_i belongs to S_i.

Remark 5.1. The power set monad $TX = \mathcal{P}(X)$ under direct images does not exactly fit into the set of examples discussed in this section since X may be infinite. However, the same formulas still hold. In particular, the sequence map is defined in exactly the same way as the sequence map for the finite power set monad.

5.5. **The representable monad.** Let E be a set. Consider the functor $TX = \mathrm{Hom}(E, -)$. This is a monad where $\epsilon_X : X \to \mathrm{Hom}(E, X)$ embeds as the constant functions, and μ_X uses pullback along the diagonal $\Delta : E \to E \times E$:

$$T^2X = \mathrm{Hom}(E, \mathrm{Hom}(E, X)) \simeq \mathrm{Hom}(E \times E, X) \xrightarrow{\Delta^*} \mathrm{Hom}(E, X) = TX.$$

To describe the Kleisli composition, we make use of the $(\times, \mathrm{Hom})$ adjunction:

$$\begin{aligned} TX \times \mathrm{Hom}(X, TY) &= \mathrm{Hom}(E, X) \times \mathrm{Hom}(X, \mathrm{Hom}(E, Y)) \\ &\simeq \mathrm{Hom}(E, X) \times \mathrm{Hom}(E, \mathrm{Hom}(X, Y)) \\ &\simeq \mathrm{Hom}(E, X \times \mathrm{Hom}(X, Y)) \\ &\to \mathrm{Hom}(E, Y) \end{aligned}$$

where the third line uses the fact that a morphism into a product is the same as a pair of morphisms into each factor, and the last map is post-composition with the evaluation map $X \times \mathrm{Hom}(X, Y) \to Y$. The map $T^{(2)}$ can be described as:

$$\begin{gathered} T^{(2)} : \mathrm{Hom}(X \times Y, Z) \to \mathrm{Hom}(\mathrm{Hom}(E, X) \times \mathrm{Hom}(E, Y), \mathrm{Hom}(E, Z)) \\ f \mapsto [(\phi, \psi) \mapsto f \circ \Delta^*(\phi \times \psi)] \end{gathered}$$

So that $T^{(2)} f(\phi, \psi)$ sends e to $f(\phi(e), \psi(e))$. The sequence map falls out of properties of how product and coproducts interact with Hom sets:

$$\begin{aligned} \mathsf{seq}_X : [\mathrm{Hom}(E, X)] &= \sum_{n=0}^{\infty} \mathrm{Hom}(E, X)^n = \sum_{n=0}^{\infty} \mathrm{Hom}(E, X^n) \\ &\to \mathrm{Hom}(E, \sum_{n=0}^{\infty} X^n) = \mathrm{Hom}(E, [X]) \end{aligned}$$

More explicitly:

$$\mathsf{seq}_X([\phi_1, \ldots, \phi_n])(e) = [\phi_1(e), \ldots, \phi_n(e)] \in [X]$$

where each ϕ_i is a morphism $E \to X$.

6. Exercises

(1) Let $F : \mathcal{C} \to \mathcal{D}$ be a functor. Show that the application of F to morphisms defines a natural transformation $\mathrm{Hom}(-,-) \to \mathrm{Hom}(F-, F-)$ of functors $\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathsf{Set}$.

(2) Let $F : \mathcal{C} \to \mathcal{D}$ be a functor, and suppose both $\mathcal{C}$ and $\mathcal{D}$ have products and a terminal object.

(a) Define a natural transformation:

$$\mathsf{cart} : F \times (- \times -) \to (- \times_{F(1)} -) \circ (F \times F)$$

of functors $\mathcal{C} \times \mathcal{C} \to \mathcal{D}$. (Hint: use the projection maps $\pi_1 : X \times Y \to X$ and $\pi_2 : X \times Y \to Y$.)

(b) For a monad T, argue that $\mathsf{cart}_{X,Y} \circ \epsilon_{X \times Y} = \epsilon_X \times \epsilon_Y$ as morphisms from $X \times Y$ to $TX \times TY$.

(c) The functor F is said to be *Cartesian* if cart is a natural isomorphism. Argue that the list endofunctor is Cartesian, but the bag endofunctor is not.

(3) Show that requiring the associativity axiom for a monad T is equivalent to requiring the multiplication in the Kleisli category to be associative. Similarly, show that the unit axiom for a monad T is equivalent to the identity properties of $\epsilon_X \in \mathcal{C}_T(X, X)$ in the Kleisli category.

(4) Given a monad T, recursively define $\mu^{(n)} : T^{n+1}X \to TX$ as:

$$\begin{aligned} \mu^{(0)} &= \mathrm{id}_{TX} \\ \mu^{(n)} &= \mu^{(n-1)} \circ \mu_{T^{n-1}X} \qquad \text{for } n \geq 1 \end{aligned}$$

Show that $\mu^{(n)} = \mu^{(n-1)} \circ T\mu_{T^{n-2}X}$ for $n \geq 2$.

(5) Let R be a semiring and let $T = \mathrm{Hom}_{\mathrm{fin}}(-, R)$. Define a map:

$$\begin{aligned} m^{(n)} : X \times TX \times T^2X \times \cdots \times T^nX &\to R \\ (\phi_0, \phi_1, \ldots, \phi_n) &\mapsto \prod_{i=1}^{n} \phi_i(\phi_{i-1}) \end{aligned}$$

Recall the map $\mu^{(n)} : T^{n+1}X \to TX$. Show that:

$$\mu^{(n)}(\Theta)(x) = \sum_{\vec{\phi}} m^{(n+1)}(\Theta, \vec{\phi}, x)$$

where the sum is over all $\vec{\phi} = (\phi_1, \ldots, \phi_n) \in TX \times \cdots \times T^nX$ and $(\Theta, \vec{\phi}, x) = (\Theta, \phi_1, \ldots, \phi_n, x)$.

(6) Show that the nil_X and cons_X fit into a natural isomorphism $\langle \mathsf{nil}, \mathsf{cons} \rangle : P + \mathrm{id} \times \mathsf{List} \xrightarrow{\sim} \mathsf{List}$, where $P : \mathsf{Set} \to \mathsf{Set}$ is the functor sending each set to the one-point set 1 (the final object in Set), and id is the identity functor.

(7) Let T be a monad. For $f \in \mathrm{Hom}(X \times Y, Z)$ prove the following equality of functions $X \times Y \to TZ$:

$$T^{(2)} f \circ (\epsilon_X \times \epsilon_Y) = \epsilon_Z \circ f$$

(8) Let T be a monad. Recall the Cartesian map from Exercise 2; we compose it with the obvious inclusion:

$$T(X \times Y) \xrightarrow{\mathsf{cart}_{X,Y}} TX \times_{T1} TY \xhookrightarrow{\mathrm{inc}} TX \times TY$$

Applying $T^{(2)}_{X,Y}$ and pulling back along this composition, we obtain:

$$\mathrm{Hom}(X \times Y, Z) \xrightarrow{T^{(2)}_{X,Y}} \mathrm{Hom}(TX \times TY, TZ) \xrightarrow{(\mathrm{inc} \circ \mathsf{cart}_{X,Y})^*} \mathrm{Hom}(T(X \times Y), TZ)$$

Prove that, in general, this composition is NOT just the functor T applied to morphisms. For example, take T to be list monad.

(9) Let T be a monad, and let $p \in TX$ for some set X. Show that $\mathsf{seq}([p]) = T[x \to [x]](p)$. In other words, we apply T to the morphism $X \to [X]$ that takes each element to the singleton list, and evaluate at $p \in TX$. Furthermore, show that $\mathsf{seq}([\epsilon_X x]) = \epsilon_{[X]}[x]$.

(10) Let M be a monoid. Argue that $X \mapsto X \times M$ is a monad. Compute the Kleisli composition and sequence maps.

(11) Implement the sequence function using the identity

$$T^{(2)}(f)(p, q) = U_T(x \mapsto Tf(x, -)(q))(p).$$

(12) Let F be an endofunctor on the category of sets.

(a) Define a natural transformation

$$\mathrm{Hom}(X \times Y, Z) \longrightarrow \mathrm{Hom}(FX \times FY, F^2 Z)$$

of functors $\mathsf{Set}^{\mathrm{op}} \times \mathsf{Set}^{\mathrm{op}} \times \mathsf{Set} \to \mathsf{Set}$. In fact there are two canonical choices; select the one that "processes" the left factor first, so that the first step in the composition is the isomorphism $\mathrm{Hom}(X \times Y, Z) \xrightarrow{\sim} \mathrm{Hom}(Y, \mathrm{Hom}(X, Z))$.

(b) Generalize by induction to a natural transformation

$$\mathrm{Hom}\,(X_1 \times \cdots \times X_n, Z) \to \mathrm{Hom}\,(FX_1 \times \cdots \times FX_n, F^n Z)$$

of functors $(\mathsf{Set}^{\mathrm{op}})^n \times \mathsf{Set} \to \mathsf{Set}$. (Again, there are choices, but proceed left to right.)

(c) In particular, considering the diagonal in $(\mathsf{Set}^{\mathrm{op}})^n$, we have a natural transformation
$$\mathrm{Hom}\,(X^n, Z) \to \mathrm{Hom}\,((FX)^n, F^n Z)$$
of functors $\mathsf{Set}^{\mathrm{op}} \times \mathsf{Set} \to \mathsf{Set}$.

(d) Now suppose $F = T$ is a monad. Using part (c) and $\mu : T^2 \to T$ define a natural transformation:
$$T^{(n)}_{X,Z} : \mathrm{Hom}(X^n, Z) \to \mathrm{Hom}((TX)^n, TZ)$$
with $T^{(0)}_Z : Z \to TZ$ being the unit, $T^{(1)}_{X,Z} : \mathrm{Hom}(X, Z) \to \mathrm{Hom}(TX, TZ)$ being the functor T applied to morphisms, and $T^{(2)}$ coinciding with the definition in Section 3.

(e) Use the $T^{(n)}$ to define a natural transformation:
$$T^{(*)} : \mathrm{Hom}([X], Z) \to \mathrm{Hom}([TX], TZ)$$
Taking $Z = [X]$, argue that $\mathrm{id}_{[X]}$ is mapped to the sequence map seq_X.

(f) Implement $T^{(*)}$ as a function in Haskell via:

```
star :: Monad m => ([a]-> b) -> [m a] -> m b
star f [] = return (f [])
star f (p:ps) = p >>= \x -> star (\ys -> f (x:ys)) ps
```

Argue that the function `star f` is equal to `liftM f . sequence`. In particular, `star id` is equal to `sequence`.

References

[Mil19] Bartosz Milewski, *Category Theory for Programmers by Bartosz Milewski | Blurb Books*, 2019.
[oH] University of Helsinki, *Haskell MOOC*. Accessed: 2025-08-13.
[Rie17] Emily Riehl, *Category Theory in Context*, Courier Dover Publications, 2017.

Appendix A. An alternative characterization of monads

Consider the following data:

- An endofunctor $T : \mathsf{Set} \to \mathsf{Set}$.
- A natural transformation $\epsilon : 1 \to T$ of endofunctors of Set.
- A natural transformation $p : \mathrm{Hom}(-, T-) \to \mathrm{Hom}(T-, T-)$ of functors $\mathsf{Set}^{\mathrm{op}} \times \mathsf{Set} \to \mathsf{Set}$.

For any X, set $\mu_X = p_{TX,X}(\mathrm{id}_{TX}) : T^2X \to TX$. It is easy to see that $\mu : T^2 \to T$ is a natural transformation.

Proposition A.1. *The data (T, ϵ, μ) is a monad if and only if the following identities hold:*

$$p_{XY}(g) \circ \epsilon_X = g \qquad p_{XX}(\epsilon_X) = \mathrm{id}_{TX} \qquad p_{YZ}(g_2) \circ p_{XY}(g_1) = p_{XZ}(p_{YZ}(g_2) \circ g_1)$$

where $g = g_1 : X \to TY$ and $g_2 : Y \to TZ$.

Sketch of proof. The forward implication follows from Lemma 2.1 above, setting p to be the natural transformation on morphisms induced by U_T. For the opposite implication, one must show that the associativity and unit axioms hold. The easiest is the left unit axiom, which follows immediately from definitions and hypotheses:

$$\mu_X \circ \epsilon_{TX} = p_{TX,X}(\mathrm{id}_{TX}) \circ \epsilon_{TX} = \mathrm{id}_{TX}$$

The verification of the right unit axiom is:

$$\begin{aligned} \mu_X \circ T\epsilon_X &= p_{TX,X}(\mathrm{id}_{TX}) \circ T\epsilon_X = p_{X,X}(\mathrm{id}_{TX} \circ \epsilon_X) \\ &= p_{X,X}(\epsilon_X) = \mathrm{id}_{TX} \end{aligned}$$

where the first equality follows from definitions, the second from the fact that p is a natural transformation, and the fourth from the hypotheses. Finally, for associativity:

$$\begin{aligned} \mu_X \circ T\mu_X &= p_{TX,X}(\mathrm{id}_{TX}) \circ T\mu_X = p_{T^2X,X}(\mathrm{id}_{TX} \circ \mu_X) = p_{T^2X,X}(\mu_X) \\ &= p_{T^2X,X}(\mu_X \circ \mathrm{id}_{T^2X}) = \mu_X \circ p_{T^2X,TX}(\mathrm{id}_{T^2X}) = \mu_X \circ \mu_{TX} \end{aligned}$$

where we have twice used the fact that p is a natural transformation. □

Appendix B. The generalized sequence map

B.1. **Introduction.** In this appendix, we give a sketch of a generalization of the sequence map beyond the List inductive type. We assume familiarity with F-algebras and initial algebras.

To warm-up, recall that $\mathsf{List}(X)$ is the initial algebra of the endofunctor $F_X : S \mapsto 1 + X \times S$. Another example to consider is that of binary trees. The set $\mathsf{BTree}(X)$ of binary trees with nodes labeled by X is the initial algebra of the endofunctor $F_X : S \mapsto 1 + X \times S^2$. In both cases, we have the assignment $X \to F_X$ constittutes a functor $\mathsf{Set} \to \mathrm{End}(\mathsf{Set})$.

B.2. **Statement.** Now fix a non-negative integer $d \geq 1$ and consider a functor:

$$\mathsf{Set}^d \to \mathrm{End}(\mathsf{Set}), \qquad \mathbf{X} = (X_1, \ldots, X_n) \mapsto F_{\mathbf{X}}$$

that takes a tuple of sets to an endofunctor of Set. We make the following assumptions:

> *Assumption 1*: The endofunctor $F_{\mathbf{X}}$ has an initial algebra for every $\mathbf{X}$; denote it as $\eta(F; X_1, \ldots, X_d)$. This gives a functor $\eta(F; -) : \mathsf{Set}^d \to \mathsf{Set}$.
>
> *Assumption 2*: For each $\mathbf{X}$ and S, the set $F_{\mathbf{X}}(S)$ is a coproduct of terms of the form $S^{m_0} \times X_1^{m_1} \times \cdots \times X_d^{m_d}$ for some $m_i \in \mathbb{N}$.

Then we can define a generalized sequence natural transformation:

$$\mathsf{seq} : \eta(F; -) \circ T^d \to T\eta(F; -)$$

of functors $\mathsf{Set}^d \to \mathsf{Set}$.

B.3. **Steps in construction.** Here is a sketch of the steps in defining this natural transformation:

- First, observe that the $T^{(2)}$ construction generalizes to natural maps:

  $$T^{(n)} = T^{(n)}_{X_1,\ldots,X_n,Z} : \mathrm{Hom}(X_1 \times \cdots \times X_n, Z) \to \mathrm{Hom}(TX_1 \times \cdots \times TX_n, TZ)$$

  where $T^{(0)} = \epsilon_Z$ and $T^{(1)} : \mathrm{Hom}(X, Z) \to \mathrm{Hom}(TX, TZ)$ is simply the endofunctor T applied to morphisms[3]. See Exercise 12.

- Recall Assumption 1. Make the following abbreviations of the initial algebras:

  $$\eta(\mathbf{X}) := \eta(F; X_1, \ldots, X_n) \qquad \eta(T\mathbf{X}) := \eta(F; TX_1, \ldots, TX_n).$$

  Initial algebras are fixed points, so we have isomorphisms

  $$c_{\mathbf{X}} : F_{\mathbf{X}}(\eta(\mathbf{X})) \xrightarrow{\sim} \eta(\mathbf{X}) \qquad c_{T\mathbf{X}} : F_{T\mathbf{X}}(\eta(T\mathbf{X})) \xrightarrow{\sim} \eta(T\mathbf{X})$$

- Recall Assumption 2. Let

  $$c'_{\mathbf{X}} : \eta(F; \mathbf{X})^{m_0} \times X^{m_1} \times \cdots \times X^{m_d} \to \eta(F; \mathbf{X})$$

  be the restriction of $c_{\mathbf{X}}$ to a cofactor. We have:

  $$T^{(m)}(c'_{\mathbf{X}}) : T\eta(F; \mathbf{X})^{m_0} \times (TX)^{m_1} \times \cdots \times (TX)^{m_d} \to T\eta(F; \mathbf{X})$$

  where $m = \sum_{i=0}^{d} m_i$.

- Observe that the domain of each $T^{(m)}(c'_{\mathbf{X}})$ is a cofactor of $F_{T\mathbf{X}}(T\eta(\mathbf{X}))$. Hence, proceeding cofactor by cofactor, we obtain a map:

  $$T^{(*)}(c_{\mathbf{X}}) : F_{T\mathbf{X}}(T\eta(\mathbf{X})) \to T\eta(\mathbf{X})$$

[3]Referring to Diagram 3.3, we have that $\epsilon_{[X]} \circ \mathsf{nil}_X = T^{(0)}_{[X]}(\mathsf{nil}_X)$.

This is precisely an $F_{T\mathbf{X}}$-algebra structure on $T\eta(\mathbf{X})$. The initiality of $\eta(T\mathbf{X})$ now provides a map (referred to as a catamorphism):

$$\eta(T\mathbf{X}) \to T\eta(\mathbf{X})$$

which is what we take as the generalized sequence map.

Thus, we have a commutative diagram:

$$\begin{array}{ccc} F_{T\mathbf{X}}(\eta(T\mathbf{X})) & \xrightarrow{F_{T\mathbf{X}}(\mathsf{seq}_X)} & F_{T\mathbf{X}}(T\eta(\mathbf{X})) \\ \Big\downarrow{\scriptstyle c_{TX}} & & \Big\downarrow{\scriptstyle T^{(*)}(c_{\mathbf{X}})} \\ \eta(T\mathbf{X}) & \xrightarrow{\mathrm{seq}_X} & T\eta(\mathbf{X}) \end{array}$$

where $T^{(*)}(c_{\mathbf{X}}) : F_{T\mathbf{X}}(T\eta(\mathbf{X})) \to T\eta(\mathbf{X})$ is our notation for the map obtained by applying the various $T^{(m)}$ on the restriction of $c_{\mathbf{X}}$ to the cofactors of $F_{T\mathbf{X}}(T\eta(\mathbf{X}))$.