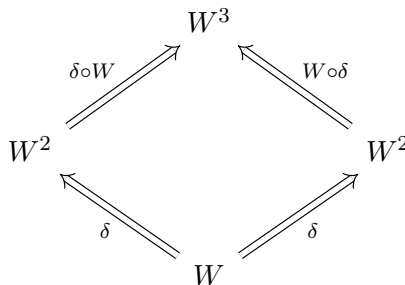


Definition 0.1. A **comonad** on a category \mathcal{C} is a triple (W, ϵ, δ)

- $W : \mathcal{C} \rightarrow \mathcal{C}$ (an endo-functor)
- $\epsilon : W \Rightarrow \mathbf{1}_{\mathcal{C}}$ (the *extract* natural transformation)
- $\delta : W \Rightarrow W^2$ (the *duplicate* natural transformation)

such that

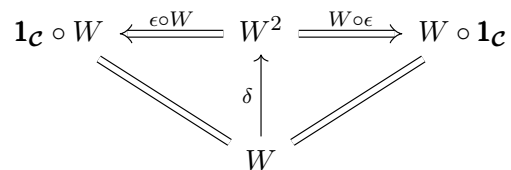
1. the diagram



commutes, that is

$$(\delta \star \mathbf{1}_W) \circ \delta = (\mathbf{1}_W \star \delta) \circ \delta$$

2. the diagram



commutes.

a comonad is just a comonoid in the category of endo-functors

Example 0.1. In Haskell a comonad is defined as

```
class (Functor w) => Comonad where
  duplicate :: w a -> w (w a) -- co-join
  extract   :: w a -> a       -- co-return
```

we can define the extend and bird operators as

```
-- extend (co-bind)
extend :: (w a -> b) -> w a -> w b
extend f = (fmap f) . duplicate

-- bird (co-fish)
(=>=) :: (w a -> b) -> (w b -> c) -> (w a -> c)
f =>= g = g . (extend f)
```

and =>= is a co-Kleisli arrow.

Example 0.2. *In Haskell the Stream functor is a comonad*

```
data Stream a = Cons a (Stream a)

extract :: w a -> a
extract (Cons a _) = a

duplicate :: w a -> w (w a)
duplicate Cons a as = Cons (Stream a as) (duplicate as)
```

Example 0.3. *Take the moving average over a stream*

```
sumN :: Num a => Int n -> Stream a -> a
sumN n (Cons a as) = if n==0
  then 0
  else a + (sumN (n-1) as)

avgN :: Fractional a => Int n -> Stream a -> a
avgN n as = (sumN n as) / (fromIntegral n)
```

avgN is local, it only looks to n neighbors. If we partially apply avgN it becomes a co-Kleisli arrow then we can use extend (avgN n) as which replaces every element of the stream with its moving average.

See [1] Ch.23 and also

- Category Theory II 7.1: Comonads - YouTube
- Category Theory II 7.2: Comonads Categorically and Examples - YouTube

[1] B. Milewski, *Category Theory for Programmers* (2019).
