

# Because it's there

William Troiani

December 4, 2022

## 1 Provenance

Consider a set  $S$ . One can define the **complex vector space freely generated by  $S$**   $\mathbb{C}[S]$  in the following way:

- The elements of  $\mathbb{C}[S]$  are finite formal sums

$$z_1 s_1 + \dots + z_n s_n \tag{1}$$

where  $n > 0$ ,  $z_i \in \mathbb{C}$ , and  $s_i \in S$  for all  $i = 1, \dots, n$ .

- Scalar multiplication is defined as follows

$$\begin{aligned} \mathbb{C} \times \mathbb{C}[S] &\longrightarrow \mathbb{C}[S] \\ (z, z_1 s_1 + \dots + z_n s_n) &\longmapsto z z_1 s_1 + \dots + z z_n s_n \end{aligned}$$

and addition is performed by “gathering like terms”.

This defines a complex vector space. There is an obvious map  $S \longrightarrow \mathbb{C}[S]$  and the elements in the image of this map (which we identify with  $S$ ) form a basis for  $\mathbb{C}[S]$ . In fact, quite a lot of the structure on  $\mathbb{C}[S]$  is determined by the set  $S$ . For instance, let  $V$  be an arbitrary complex vector space, and consider a morphism  $\varphi : \mathbb{C}[S] \longrightarrow V$ . By linearity, we have following calculation for an arbitrary vector  $z_1 s_1 + \dots + z_n s_n \in \mathbb{C}[S]$

$$\varphi(z_1 s_1 + \dots + z_n s_n) = z_1 \varphi(s_1) + \dots + z_n \varphi(s_n) \tag{2}$$

and so  $\varphi$  is determined by the set  $\varphi(S)$ . In other words, if we let  $UV$  denote the underlying vector space of  $V$ , then the map  $\varphi$  is determined by its restriction  $\varphi \upharpoonright_S : S \longrightarrow UV$ . Moreover, *every* morphism  $\mathbb{C}[S] \longrightarrow V$  can be given in this way. So we have:

**Proposition 1.0.1.** *For any set  $S$  and any complex vector space  $V$ , there is a bijection:*

$$\mathrm{Hom}_{\mathbb{C}\mathrm{Vect}}(\mathbb{C}[S], V) \cong \mathrm{Hom}_{\underline{\mathrm{Set}}}(S, UV) \quad (3)$$

In fact, more can be said. If  $f : S \rightarrow T$  is a function of sets, and  $\varphi : V \rightarrow W$  a linear transformation, then there exists a pair of commuting diagrams:

$$\begin{array}{ccc} \mathrm{Hom}(\mathbb{C}[S], V) & \longrightarrow & \mathrm{Hom}(S, UV) & \mathrm{Hom}(\mathbb{C}[S], V) & \longrightarrow & \mathrm{Hom}(S, UV) \\ \uparrow \scriptstyle{-\circ\mathbb{C}[f]} & & \uparrow \scriptstyle{-\circ f} & \downarrow \scriptstyle{\varphi\circ-} & & \downarrow \scriptstyle{U\varphi\circ-} \\ \mathrm{Hom}(\mathbb{C}[T], V) & \longrightarrow & \mathrm{Hom}(T, UV) & \mathrm{Hom}(\mathbb{C}[S], W) & \longrightarrow & \mathrm{Hom}(S, UW) \end{array} \quad (4)$$

That is, (3) is *natural*.

In this discussion we have focussed on vector spaces, but similar constructions exist: the group freely generated by  $S$ , the monoid freely generated by  $S$ , the category freely generated by a directed graph, etc. Thus, we make a general definition, *because it's there*.

**Definition 1.0.2.** Let  $F : \mathcal{C} \rightarrow \mathcal{D}$  and  $G : \mathcal{D} \rightarrow \mathcal{C}$  be a pair of functors. The functor  $F$  is **left adjoint to  $G$**  if for all  $Y \in \mathcal{D}$  and all  $X \in \mathcal{C}$  there exists a natural bijection of homsets:

$$\mathrm{Hom}_{\mathcal{C}}(GY, X) \cong \mathrm{Hom}_{\mathcal{D}}(Y, FX)$$

We say that  $(F, G)$  are an **adjoint pair** and write  $F \dashv G$ .

## 2 Adjunctions

If we take  $\mathcal{C} = \mathbb{C}\mathrm{Vect}$ ,  $\mathcal{D} = \underline{\mathrm{Set}}$ ,  $F = U$  the **forgetful functor** which maps a vector space to its underlying set and a linear transformation to itself but where we forget the fact that it is linear, and  $G = \mathbb{C}[_]$  the **free functor** which maps a set  $S$  to the complex vector space freely generated by  $S$  and a function of sets to its induced linear transformation, then we recover the example given in the introduction and see that the forgetful functor is left adjoint to the free functor.

**Example 2.0.1.** Find a left adjoint to the forgetful functor  $U : \underline{\mathrm{Set}}_* \rightarrow \underline{\mathrm{Set}}$  which maps a pointed set  $(X, x)$  (ie, a set  $X$  along with an element  $x \in X$ ) to the set  $X$ .

### 3 Monads

An adjunction is a pair of functors  $F : \mathcal{C} \rightarrow \mathcal{D}, G : \mathcal{D} \rightarrow \mathcal{C}$ , which means that we have two distinct sides, the side focussed on  $\mathcal{C}$ , and the side focussed on  $\mathcal{D}$ . What amount of this adjunction is visible on only one side of this adjunction? For instance, the category  $\mathcal{C}$  can “see” the composite  $GF : \mathcal{C} \rightarrow \mathcal{C}$ . What else? Well, returning to the defining bijection of an adjunction (assuming  $F \dashv G$ ):

$$\text{Hom}_{\mathcal{D}}(FX, Y) \cong \text{Hom}_{\mathcal{C}}(X, GY) \quad (5)$$

If we put  $Y = FX$ , then we get

$$\text{Hom}_{\mathcal{D}}(FX, FX) \cong \text{Hom}_{\mathcal{C}}(X, GFX) \quad (6)$$

where the left hand side has a special element, the identity  $\text{id}_{FX}$ . The image of this element under the bijection is a special element  $\eta_X : X \rightarrow GFX$ . In fact, the collection of these fit into a natural transformation

$$\eta : \text{id}_{\mathcal{C}} \Rightarrow GF \quad (7)$$

between endofunctors on  $\mathcal{C}$ . On the other hand, we could also take  $X = GY$  in the above adjunction, and obtain a natural transformation

$$\epsilon : FG \Rightarrow \text{id}_{\mathcal{D}} \quad (8)$$

However this is between endofunctors on  $\mathcal{D}$ , and so is not visible from the perspective of  $\mathcal{C}$ . However, for any  $X \in \mathcal{C}$  we could consider the image under  $G$  of the morphism  $FGFX$  and obtain a natural transformation

$$G\epsilon_F : GFGF \Rightarrow GF \quad (9)$$

between a pair of endofunctors on  $\mathcal{C}$ . Taking  $M = GF$  and  $\mu = G\epsilon_F$  we have in summary the following data:

- A functor  $M$ .
- A pair of natural transformations.

$$\eta : \text{id}_{\mathcal{C}} \Rightarrow M \quad \mu : M^2 \Rightarrow M \quad (10)$$

This defines the data of a **monad** on  $\mathcal{C}$ , however this data actually obeys certain axioms due to naturality of the defining bijection involved in an adjunction pair. Once one has done the work involved in unravelling this structure, we arrive at the following, *because it's there*.

**Definition 3.0.1.** A **monad** is the data of a functor  $M : \mathcal{C} \rightarrow \mathcal{C}$  along with a pair of natural transformations  $\eta : \text{id}_{\mathcal{C}} \Rightarrow M, \mu : T^2 \Rightarrow T$  such that the following diagrams commute.

$$\begin{array}{ccc}
 T^3 & \xrightarrow{T\mu} & T^2 \\
 \mu_T \downarrow & & \downarrow \mu \\
 T^2 & \xrightarrow{\mu} & T
 \end{array}
 \quad
 \begin{array}{ccc}
 T & \xrightarrow{T\eta} & T^2 & \xleftarrow{\eta_T} & T \\
 & \searrow \text{id} & \downarrow \mu & \swarrow \text{id} & \\
 & & T & & 
 \end{array}
 \tag{11}$$

**Exercise 3.0.2** (Hard exercise). Think about the data of an adjunction pair from the other side of the adjunction to invent the definition of a *comonad*.

**Example 3.0.3.** Let  $T : \text{Set} \rightarrow \text{Set}$  act as

$$TA = \coprod_{n \geq 0} A^n \tag{12}$$

Let  $\eta_A : A \rightarrow TA$  be defined by  $a \mapsto (a)$ , and let  $\mu$  be concatenation. Then  $(T, \eta, \mu)$  is a monad.

**Example 3.0.4.** Let  $T : \text{Set} \rightarrow \text{Set}$  act as

$$TA = A \coprod \{\perp\} \tag{13}$$

where  $\eta_A : A \rightarrow TA$  maps  $a \mapsto a$ , and  $\mu_A : T^2A \rightarrow TA$  maps both copies of  $\perp$  to  $\perp$  and acts identically on the remaining elements of  $T^2A$ .

Monads are important in mathematics, but they have also been used extensively in computer science. The previous two examples give a hint as to how. They are very helpful for modelling side effects of programs! In functional computer languages where side effects are not possible, the use of monads becomes imperative. Next lecture we will see more precisely how monads have been used both in the mathematical modelling of notions of computation, and also how they have been used practically in the context of a real programming language, Haskell.