ADJOINT FUNCTOR THEOREM

BARTOSZ MILEWSKI

One of the tropes of detective movies is the almost miraculous ability to reconstruct an image from a blurry photograph. You just scan the picture, say ”enhance!”, and voila, the face of the suspect or the registration number of their car appear on your computer screen.

Figure 1. Computer, enhance!

With constant improvements in deep learning, we might eventually get there. In category theory, though, we do this all the time. We recover lost information. The procedure is based on the basic tenet of category theory: an object is defined by its interactions with the rest of the world. This is the basis of all universal constructions, the Yoneda lemma, Grothendieck fibration, Kan extensions, and practically everything else.

An iconic example is the construction of the left adjoint to a given functor, and that's what we are going to study here. But first let me explain why I decided to pick this subject, and how it's related to programming. I wanted to write a blog post about CPS (continuation passing style) and defunctionalization, and I stumbled upon an article in nLab that related defunctionalization to Freyd's Adjoint Functor Theorem; in particular to the Solution Set Condition. Such an unexpected connection peaked my interest and I decided to dig deeper into it.

1. Adjunctions

Consider a functor R from some category $\mathcal{D}$ to another category $\mathcal{C}$.

$$R\colon D \to C$$

A functor, in general, loses some data, so it's normally impossible to invert it. It produces a ”blurry” image of $\mathcal{D}$ inside $\mathcal{C}$. Its left adjoint is a functor from $\mathcal{C}$ to $\mathcal{D}$

$$L\colon C \to D$$

that attempts to reconstruct lost information, to the best of its ability. Often the functor R is *forgetful*, which means that it purposefully forgets some information. Its left adjoint is then called *free*, because it freely ad-libs the forgotten information.

Of course it's not always possible, but under certain conditions such left adjoint exists. These conditions are spelled out in the Freyd's General Adjoint Functor Theorem.

To understand them, we have to talk a little about size issues.

2. Size issues

A lot of interesting categories are large. It means that there are so many objects in the category that they don't even form a set. The category of all sets, for instance, is large (there is no set of all sets). It's also possible that morphisms between two objects don't form a set.

A category in which objects form a set is called *small*, and a category in which hom-sets are sets is called *locally small.*

A lot of complexities in Freyd's theorem are related to size issues, so it's important to precisely spell out all the assumptions.

We assume that the source of the functor R, the category $\mathcal{D}$, is locally small. It must also be small-complete, that is, every small diagram in $\mathcal{D}$ must have a limit. (A small diagram is a functor from a small category.) We also want the functor R to be continuous, that is, to preserve all small limits.

If it weren't for size issues, this would be enough to guarantee the existence of the left adjoint, and we'll first sketch the proof for this simplified case. In the general case, there is one more condition, the Solution Set Condition, which we'll discuss later.

3. Left adjoint and the comma category

Here's the problem we are trying to solve. We have a functor R that maps objects and morphisms from $\mathcal{D}$ to $\mathcal{C}$. We want to define another functor L that goes in the opposite direction. We're not looking for the inverse, so we're not expecting the composition of this functor with R to be identity, but we want it to be *related* to identity by two natural transformations called unit and counit. Their components are, respectively:

$$\eta_c : c \to RLc$$

$$\epsilon_d : LRd \to d$$

and, as long as they satisfy some additional triangle identities, they will establish the adjunction $L \dashv R$.

We are going to define L point-wise, so let's pick an object c in $\mathcal{C}$ and try to propagate it back to $\mathcal{D}$. To do that, we have to gather as much information about c as possible. We will propagate all this information back to $\mathcal{D}$ and find an object in $\mathcal{D}$ that "looks the same." Think of this as creating a hologram of c and shipping it back to $\mathcal{D}$.

All information about c is encoded in morphisms so, in order to generate our hologram, we'll gather all morphisms that originate in c. These morphisms form a category called the *coslice* category c/C.

The objects in c/C are pairs $(x, f\colon c \to x)$. In other words, these are all the arrows that emanate from c, indexed by their target objects x. But what really defines the structure of this category are morphisms between these arrows. A morphism in c/C from (x, f) to (y, g) is a morphism $h\colon x \to y$ that makes the following triangle commute:

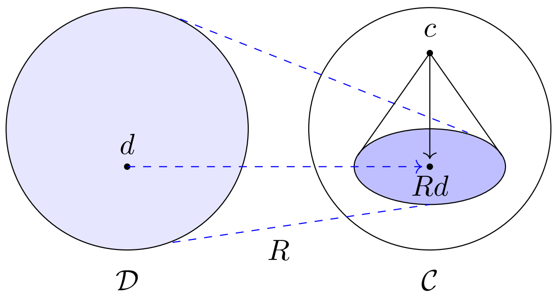
$$\begin{tikzcd} & c \arrow[dl, "f"'] \arrow[dr, "g"] & \\ x \arrow[rr, "h"] & & y \end{tikzcd}$$

We now have complete information about c encoded in the slice category, but we have no way to propagate it back to $\mathcal{D}$. This is because, in general, the image of $\mathcal{D}$ doesn't cover the whole of $\mathcal{C}$. Even more importantly, not all morphisms in $\mathcal{C}$ have corresponding morphisms in $\mathcal{D}$. We have to scale down our expectations, and define a partial hologram that does not capture all the information about c; only this part which can be back-propagated to $\mathcal{D}$ using the functor R. Such partial hologram is called a comma category c/R.

The objects of c/R are pairs $(d, f \colon c \to Rd)$, where d is an object in $\mathcal{D}$. In other words, these are all the arrows emanating from c whose target is in the image of R. Again, the important structure is encoded in the morphisms of c/R. These are the arrows in $\mathcal{D}$, $h \colon d \to d'$ that make the following diagram commute in $\mathcal{C}$

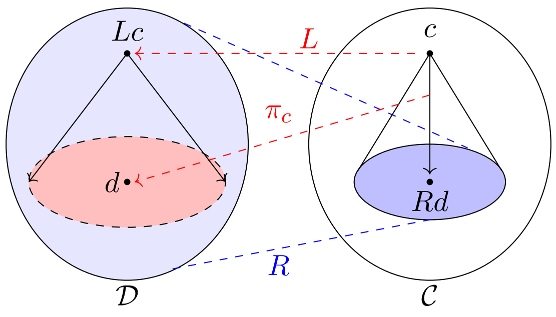
$$\begin{tikzcd}
 & c \arrow[dl, "f"'] \arrow[dr, "g"] & \\
Rd \arrow[rr, "Rh"] & & Rd'
\end{tikzcd}$$

Notice an interesting fact: we can interpret these triangles as commutation conditions in a cone whose apex is c and whose base is formed by objects and morphisms in the image of R. But not all objects or morphism in the image of R are included. Only those morphisms that make the appropriate triangle commute–and these are exactly the morphisms that satisfy the cone condition. So the comma category builds a cone in $\mathcal{C}$.



4. Constructing the limit

We can now take all this information about c that's been encoded in c/R and move it back to $\mathcal{D}$. We define a projection functor $\pi_c \colon c/R \to D$ that maps (d, f) to d, thus forgetting the morphism f. What's important, though, is that this functor keeps the information encoded in the morphisms of c/R, because these are morphisms in $\mathcal{D}$.



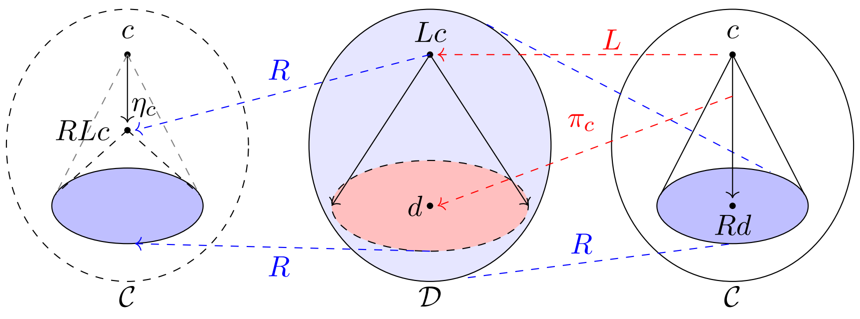
The image of π_c doesn't necessarily cover the whole of $\mathcal{D}$, because not every Rd has arrows coming from c. Similarly, only some morphisms, the ones that make the appropriate triangle in $\mathcal{C}$ commute, are picked by π_c. But those objects and morphisms that *are* in the image of π_c form a diagram in $\mathcal{C}$. This diagram is our partial hologram, and we can use it to pick an object in $\mathcal{D}$ that looks almost exactly like c. That object is the limit of this diagram. We pick the limit of this diagram as the definition of Lc: the left adjoint of R acting on c.

Here's the tricky part: we assumed that $\mathcal{D}$ was small-complete, so every small diagram had a limit; but the diagram defined by π_c is not necessarily small. Let's ignore this problem for a moment, and continue sketching the proof. We want to show that the mapping that assigns the limit of π_c to every c is left adjoint to R.

Let's see if we can define the unit of the adjunction:

$$\eta_c : c \to RLc$$

Since we have defined Lc as the limit of the diagram π_c and R preserves limits (small limits, really; but we are ignoring size problems for the moment) then RLc must be the limit of the diagram $R\pi_c$ in $\mathcal{C}$. But, as we noted before, the diagram $R\pi_c$ is exactly the base of the cone with the apex c that we used to define the comma category c/R. Since RLc is the limit of this diagram, there must be a unique morphism from any other cone to it. In particular there must be a morphism from c to it, because c is an apex of the cone defined by the comma category. And that's the morphism we'll chose as our η_c.



Incidentally, we can interpret η_c itself as an object of the comma category c/R, namely the one defined by the pair $(Lc, \eta_c \colon c \to RLc)$. In fact, this is the *initial object* in that category. If you pick any other object, say, $(d, g \colon c \to Rd)$, you can always find a morphism $h \colon Lc \to d$, which is just a leg, a projection, in the limiting cone that defines Lc. It is automatically a morphism in c/R because the following triangle commutes:

$$\begin{tikzcd}
c \arrow[d, "\eta_c"'] \arrow[dr, "g"] & \\
RLc \arrow[r, "Rh"'] & Rd
\end{tikzcd}$$

This is the triangle that defines η_c as a morphism of cones, from the top cone with the apex c, to the bottom (limiting) cone with the apex RLc. We'll use this interpretation later, when discussing the full version of the Freyd's theorem.

We can also define the counit of the adjunction. Its component at c is a morphism

$$\epsilon_d : LRd \to d$$

First, we repeat our construction starting with $c = Rd$. We define the comma category Rd/R and use π_{Rd} to create the diagram whose limit is LRd. We pick ϵ_d to be a projection in the limiting cone. We are guaranteed that d is in the base of the cone, because it's the image of $(d, id\colon Rd \to Rd)$ under π_{Rd}.

To complete this proof, one should show that the unit and counit are natural transformations and that they satisfy triangle identities.

5. End of a comma category

An interesting insight into this construction can be gained using the end calculus. In my previous post, I talked about (weighted) colimits as coends, but the same argument can be dualized to limits and ends. For instance, the comma category can be described as the following coend:

$$c/R \cong \mathcal{D} \int^{d} \mathcal{C}(c, Rd)$$

The limit of π_c over the comma category can be written as

$$\lim_{c/R} \pi_c \cong \int_{(d,f)\colon c/R} \pi_c(d, f) \cong \int_{(d,f)\colon c/R} d$$

This, in turn, can be rewritten as a weighted limit:

$$\lim^{\mathcal{C}(c,R-)} Id \cong \int_{d\colon \mathcal{D}} \mathcal{C}(c, Rd) \pitchfork d$$

If the left adjoint L exists, this can be rewritten as

$$\int_{d\colon \mathcal{D}} \mathcal{D}(Lc, d) \pitchfork d$$

which, using standard calculus of ends (see Appendix), can be shown to be isomorphic to Lc.

The pitchfork here is the power (cotensor) defined by the equation

$$\mathcal{D}\big(d', s \pitchfork d\big) \cong Set\big(s, \mathcal{D}(d', d)\big)$$

You may think of $s \pitchfork d$ as the product of s copies of the object d, where s is a set.

6. Solution set condition

So what about those pesky size issues? It's one thing to demand the existence of all small limits, and a completely different thing to demand the existence of large limits (such requirement may narrow down the available categories to preorders). Since the comma category may be too large, maybe we can cut it down to size by carefully picking up a (small) set of objects out of all objects of $\mathcal{D}$. We may take some indexing set I and construct a family d_i of objects of $\mathcal{D}$ indexed by elements of I. It doesn't have to be one family for all—we may pick a different family for every object c for which we are performing our construction.

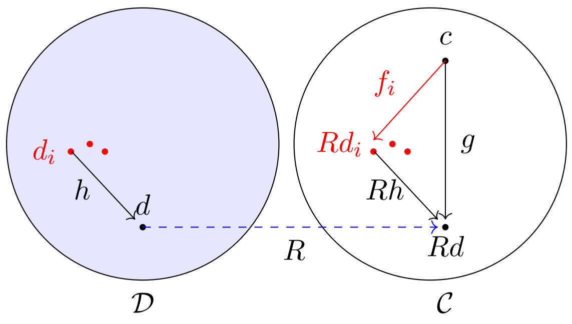
Instead of using the whole comma category c/R, we'll limit ourselves to a set of arrows $f_i\colon c \to Rd_i$. But in a comma category we also have morphisms between arrows. In fact they are the essential carriers of the structure of the comma category. Let's have another look at these morphisms.

$$\begin{tikzcd} & c \arrow[dl, "f"'] \arrow[dr, "g"] & \\ Rd \arrow[rr, "Rh"] & & Rd' \end{tikzcd}$$

This commuting condition can be re-interpreted as a factorization of g through f. It so happens that every morphism g can be trivially factorized through some f by picking $d = d'$ and $h = id_d$. But if we restrict the factors f to be members of the family f_i then not every $g \colon c \to Rd$ (for arbitrary d) can be automatically factorized. We have to demand it. That gives us the following:

Solution Set Condition: For every object c there exists a small set I with an I-indexed family of objects d_i in $\mathcal{D}$ and a family of morphisms $f_i \colon c \to Rd_i$, such that every morphism $g \colon c \to Rd$ can be factored through one of f_i. That is, there exists a morphism $h \colon d_i \to d$ such that

$$g = Rh \circ f_i$$



There is a shorthand for this statement: All comma categories c/R admit weakly initial families of objects. We'll come back to it later.

7. Freyd's theorem

We can now formulate:

Freyd's Adjoint Functor Theorem: If $\mathcal{D}$ is a locally small and small-complete category, and the functor $R \colon D \to C$ is continuous (small-limit preserving), and it satisfies the solution set condition, then R has a left adjoint.

We've seen before that the key to defining the point-wise left adjoint was to find the initial object in the comma category c/R. The problem is that this comma category may be large. So the trick is to split the proof into two parts: first defining a *weakly* initial object, and then constructing the actual initial object using equalizers. A weakly initial object has morphisms to every object in the category but, unlike its strong version, these morphisms don't have to be unique.

An even weaker notion is that of a weakly initial *set of objects*. These are objects that among themselves have arrows to every object in the category, but it's possible that no individual object has all the arrows. The solution set in Freyd's theorem is such a weakly initial set in the comma category c/R. Since we assumed that $\mathcal{C}$ is small-complete, we can take a product of these objects and show that it's weakly initial. The proof then proceeds with the construction of the initial object.

The details of the proof can be found in any category theory text or in nLab.

Next we'll see the application of these results to the problem of defunctionalization of computer programs.

8. Appendix

To show that

$$\int_d \mathcal{D}(Lc, d) \pitchfork d \cong Lc$$

it's enough to show that the hom-functors from an arbitrary object d' are isomorphic

$$\begin{aligned}
&\mathcal{D}\big(d', \int_d \mathcal{D}(Lc, d) \pitchfork d\big) \\
&\cong \int_d \mathcal{D}\big(d', \mathcal{D}(Lc, d) \pitchfork d\big) \\
&\cong \int_d Set\big(\mathcal{D}(Lc, d), \mathcal{D}(d', d)\big) \\
&\cong \mathcal{D}(d', Lc)
\end{aligned}$$

I used the continuity of the hom-functor, the definition of the power (cotensor) and the ninja Yoneda lemma.