Designing and Modeling Complex Systems with Category Theory

Daniel Sinderson

May 2024

Introduction

Category theory studies composition. By studying and abstracting notions of composition, categories manage to encapsulate a shockingly flexible and wide-reaching language for describing and working with structure of all sorts. Much like the study of sets and the set membership of elements turns out to be capable of formalizing all of mathematics, so too does category theory, except from a bird's eye point of view: the details of a particular field go out of focus and only its high level structural patterns remain. This is useful. High levels of abstraction provide high levels of generality, and a general, common language of how things are structured is useful for both organizing thought and sharing it.

Because of this generality, category theory also has some potential for reformulating difficult problems in other fields or even opening a broader class of thinking and phenomena to mathematical rigor. This is an active field of research. There are people using the tools of category to formalize and investigate everything from quantum mechanics and linguistics to philosophy and systems theory.

It's this last application to systems theory that we'll be looking more deeply at in this text. Though we're merely wandering at the edge of the work being done, we'll see how to use category theory to build the specifications for, and investigate the behaviors of, dynamical systems built from the composition of smaller, simpler systems. We'll do this by working through a case study for both discrete and differential deterministic equations. For our first case study we'll build a state machine to control the behavior of agents in a game. For our second case study we'll build a differential system for modeling a complicated food web.

With even this partial, introductory view of the subject matter it should be easy to see the promise of this approach for facilitate thinking about and designing models for complex systems.

Category Theory Basics

The first step to understanding category theory is to understand what a category is. If you're new to higher mathematics it's time to take a deep breath. This will seem like a lot.

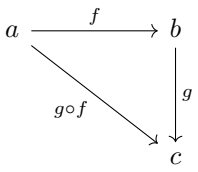
Definition 1 (Category)**.** A category $\mathcal{C}$ is defined by the following:

1. $\mathcal{C}$ contains a collection of objects $ob(\mathcal{C})$. We'll denote that an object is in a category using set notation: $c \in \mathcal{C}$.

2. For any two objects a, $b \in \mathcal{C}$ there is a collection of morphisms, or arrows, between those objects $\mathcal{C}(a, b)$ called the homset. This is short for "set of homomorphism." We'll denote an element $f \in \mathcal{C}(a, b)$ using function notation: $f : a \to b$.

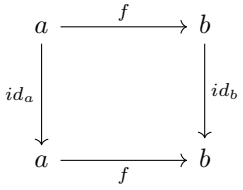
3. Every object $a \in \mathcal{C}$ has a morphism to itself $id_a : a \to a$ called its identity. This morphism doesn't do anything. It's like multiplying a number by 1.

4. For every two morphisms $f : a \to b$ and $g : b \to c$ there's a third morphism $g \circ f : a \to c$ that's their composition. The circle is the symbol for function composition and $g \circ f$ is read "g after f."

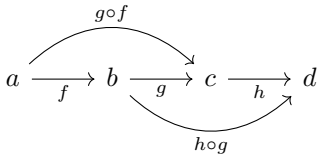


These objects and morphisms are then under the following two constraints:

1. (Unitality) Any morphism $f : a \to b$ can be composed with the identity morphisms of a and b such that $f \circ id_a = id_b \circ f = f$. This makes the following diagram commute. What this means is that both paths from a to b here are equivalent.



2. (Associativity) For any morphisms $f : a \to b$, $g : b \to c$, and $h : c \to d$, $h \circ (g \circ f) = (h \circ g) \circ f$. Since it doesn't matter what order we apply the morphisms, we write this $h \circ g \circ f$.



Let's break this down a little bit. We have a collection of objects with connections between them, and a couple of rules. We know that every object is connected to itself. And we know that if object a is connected to b, and b is connected to c, then a is connected to c through b.[1] That's it. Categories are honestly pretty simple. But from these humble beginnings we're going to climb very high.

Functors and Categories of Categories

Most mathematical structures have some way of mapping one instance of that structure to another instance. Functions map sets to sets. Linear transformations map vector spaces to other vector spaces. And group homomorphisms map groups to other groups. Since categories are a mathematical structure, like groups or vector spaces, it makes sense to ask if there exists a way to map between them. Such a map would have to preserve all of the things that make a category a category: identity morphisms and composite morphisms, and the rules of unitality and associativity. We call such a map between categories a functor.

Definition 2 (Functor)**.** A functor $F : \mathcal{C} \to \mathcal{D}$ is a map between categories $\mathcal{C}$ and $\mathcal{D}$ such that the following hold:

1. For any object $a \in \mathcal{C}$ there is an object $Fa \in \mathcal{D}$.
2. For any morphism $f : a \to b$ between objects a and b in $\mathcal{C}$ there is a morphism $Ff : Fa \to Fb$ between objects Fa and Fb in $\mathcal{D}$.
3. For all objects $a \in \mathcal{C}$ and $Fa \in \mathcal{D}$, $Fid_a = id_{Fa}$.
4. For any composition of morphisms $g \circ f$ in $\mathcal{C}$, $F(g \circ f) = Fg \circ Ff$ in $\mathcal{D}$.

There are some things to note here. The first is that, like sets and functions, the mapping of objects from one category to another is unique. Every object in the source category is mapped to exactly one object in the target category. The same goes for morphisms. The second is that identity morphisms get mapped to identity morphisms and compositions get mapped to compositions. This ensures that our two rules, unitality and associativity, will be preserved since they're expressible purely in terms of composition of morphisms, identities and otherwise.

[1] For people with some background in math or logic this might be familiar as the transitive property.

With functors in our toolbox, it's now possible to construct a category where the objects themselves are categories and the morphism between them are functors. This, along with our next step, will prove to be extremely powerful.

Avoiding Paradox

For anyone who has taken a course in logic or set theory, that last paragraph might be ringing some alarm bells. Whenever we have recursion in our structure like this, there's the opportunity for paradox to sneak in. The category of categories is no exception.

To avoid the paradox of whether the category of categories contains itself, category theorists employ formal notions of "size" to differentiate distinct universes of categories. These formalisms can get pretty technical though, so we'll be ignoring them in this paper. For us it will be enough to assume that whenenever we talk about a category whose objects are also categories, our object categories are in some way "smaller" than the category that they're a part of, and thus of a different type.[2]

Functor Categories and Natural Transformations

For our next, and last, step up into abstraction let's talk about functor categories. Functor categories are categories whose objects are functors and whose morphisms are maps called natural transformations.

Since natural transformations map functors to other functors, we know that they must preserve the essential "functor-ness" of functors. This "functor-ness" is a bit harder to pin down than the "category-ness" that functors themselves preserve. We need to see what natural transformations do to functors, but also to what they do to the objects and arrows of the categories that the functors are acting on.

Definition 3 (Natural Transformation)**.** A natural transformation $\alpha : F \Rightarrow G$ is a map between functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$ such that the following holds:

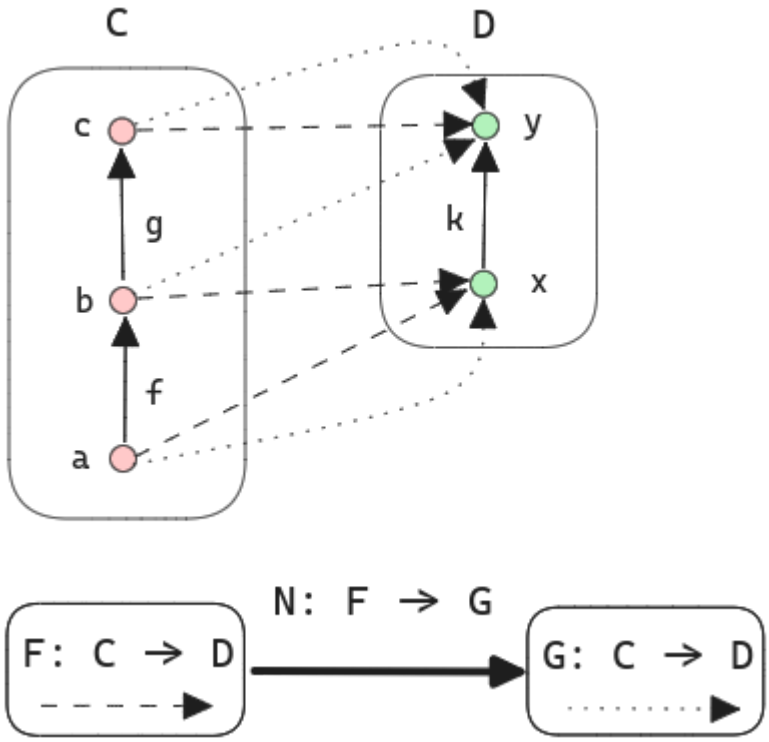
1. For each object $c \in \mathcal{C}$, there is a morphism $\alpha_c : Fc \to Gc$ in $\mathcal{D}$. This is called the c-component of α.
2. For every morphism $f : a \to b$ in $\mathcal{C}$, $\alpha_b \circ Ff = Gf \circ \alpha_a$. This is called the naturality condition, and it preserves functoriality.

The naturality condition ensures that the following diagram commutes.

$$\begin{array}{ccc} Fa & \xrightarrow{\alpha_a} & Ga \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ Fb & \xrightarrow[\alpha_b]{} & Gb \end{array}$$

[2] In general, this "smallness" will be that the category's collection of objects forms a set.

This definition is short but there's a whole extra layer of stuff to keep track of since we added a layer of abstraction. Let's get a better idea of what this definition captures by using a picture.



Here we see two simple categories, $\mathcal{C}$ and $\mathcal{D}$, and two functors F (the dashed lines) and G (the dotted lines) between them. Let's dig into this picture a bit and pick apart the functors and the natural transformation.

First, let's note all the important mappings of the functors and the component morphisms of the natural transformation.

- $Fa = x$, $Fb = x$, $Fc = y$, $Ff = id_x$, $Fg = k$, $F(g \circ f) = k$
- $Ga = x$, $Gb = y$, $Gc = y$, $Gf = k$, $Gg = id_y$, $G(g \circ f) = k$
- $N_a : Fa \to Ga$; $N_a = id_x$
- $N_b : Fb \to Gb$; $N_b = k$
- $N_c : Fc \to Gc$; $N_c = id_y$

Next, let's check that the functors preserve composition.[3]

$$F(g \circ f) = k = k \circ id_x = Fg \circ Ff \quad \checkmark$$

[3] You should be able to convince yourself that they preserve identities.

$$G(g \circ f) = k = id_y \circ k = Gg \circ Gf \quad \checkmark$$

And now let's look at the natural transformation $N : F \Rightarrow G$ and make sure that it satisfies the naturality condition for the morphisms $f, g \in \mathcal{C}$.

$$N_b \circ Ff = k \circ id_x = Gf \circ N_a \quad \checkmark$$

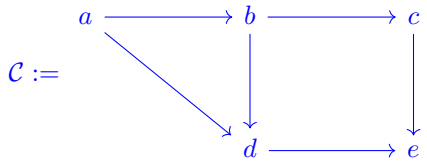
$$N_c \circ Fg = id_y \circ k = Gg \circ N_b \quad \checkmark$$

Everything is working as expected. Hopefully this helped to unpack some of the complexity behind the natural transformation definition. In a real example, our functors would be defined equationally and we'd prove these properties for arbitrary morphisms instead of proving them for each morphism individually like we did here.

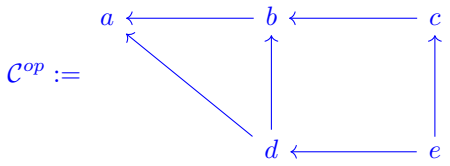
Duality

The last thing we'll cover in this section is the very simple, and very powerful, property of duality. Every category has a dual category. The objects in this dual category are exactly the same as in the original category, but every arrow is turned around: its source becomes its target and its target becomes its source. Let's look at a simple example.

Let $\mathcal{C}$ be the category below.



It's dual category then we write as $\mathcal{C}^{op}$ and define as follows.



And that's it. The reason duality turns out to be so powerful is that it essentially doubles our effectiveness. For every structure we find in the next section, we get the dual structure essentially for free. Duality also plays an important role in representability and the Yoneda embedding which are major features of category theory that we'll get to later.

Structure within Categories: Duality, Universal Properties and (Co)Limits

First though, we're going to take a look at the kinds of structure that can exist inside an individual category. We'll start with

Kinds of Morphisms

NOTE: Isomorphism in the category **Set** are bijective functions.

Definition 4 (Isomorphism)**.** We say that two objects in a category are isomorphic if there exists a morphism $f : a \to b$ for which there is another morphism $f^{-1} : b \to a$ such that $g \circ f = id_b$ and $f \circ g = id_a$. Such a morphism is called an isomorphism.

NOTE: Monomorphism in the category **Set** are injective functions.

Definition 5 (Monomorphism)**.** A morphism $f : b \to c$ is a monomorphism if for any pair of parallel morphisms $g, h : a \to b$, $f \circ g = f \circ h$ implies that $g = h$.

NOTE: Epimorphisms in the category **Set** are surjective functions.

Definition 6 (Epimorphism)**.** A morphism $f : b \to c$ is an epimorphism if for any pair of parallel morphisms $g, h : c \to d$, $g \circ f = h \circ f$ implies that $g = h$.

Universal Properties

Definition 7 (Terminal Object)**.** Let $t \in \mathcal{C}$ be given. Then t is a terminal object in $\mathcal{C}$ if the following hold:

1. For all $x \in \mathcal{C}$, there is exactly one morphism $f : x \to t$.
2. For any other object t' in $\mathcal{C}$ that meets condition (1), there exists an unique isomorphism $g : t \to t'$.

Dual to terminal objects are initial objects.

Definition 8 (Initial Object)**.** Let $i \in \mathcal{C}$ be given. Then i is a initial object in $\mathcal{C}$ if the following hold:

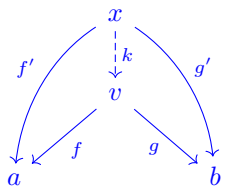
1. For all $x \in \mathcal{C}$, there is exactly one morphism $f : i \to x$.
2. For any other object i' in $\mathcal{C}$ that meets condition (1), there exists an unique isomorphism $g : i \to i'$.

In category theory there's a universal property that generalizes notions of multiplication called the product.

Definition 9 (Product)**.** Let $a, b, v \in \mathcal{C}$ and morphisms $f : v \to a$ and $g : v \to b$ be given. The object v is a product of a and b if the following holds:

1. For any other object x with morphisms $f' : x \to a$ and $g' : x \to b$, there is a unique isomorphism $k : x \to v$ such that $f \circ k = f'$ and $g \circ k = g'$.

This means the following diagram commutes:



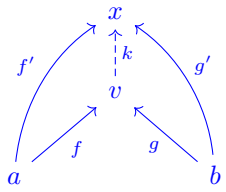
This conditions ensures that the object v is the universal or canonical object satisfying the pattern, with the morphisms of any other such object being able to be written as a sort of embellishment of the morphisms from v.

Dual to products are coproducts, which generalize the notion of addition.

Definition 10 (Coproduct)**.** Let $a, b, v \in \mathcal{C}$ and morphisms $f : a \to v$ and $g : b \to v$ be given. The object v is a coproduct of a and b if the following holds:

1. For any other object x with morphisms $f' : a \to x$ and $g' : b \to x$, there is a unique isomorphism $k : v \to x$ such that $k \circ f = f'$ and $k \circ g = g'$.

This means the following diagram commutes:

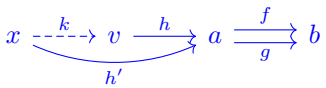


NOTE: In **Set**, equalizers pick out the subset of a set a that is the intersection of the preimages $f^{-1} : b \to a$ and $g^{-1} : b \to a$.

Definition 11 (Equalizer)**.** Let $a, b, v \in \mathcal{C}$ and let morphisms $f : a \to b$, $g : a \to b$, and $h : v \to a$ be given. The object v is an equalizer if the following hold:

1. $f \circ h = g \circ h$.
2. For any other object $x \in \mathcal{C}$ with morphism $h' : x \to a$ such that $f \circ h' = g \circ h'$, there exists a unique morphism $k : x \to v$ such that $h' = h \circ k$.

This means the following diagram commutes:

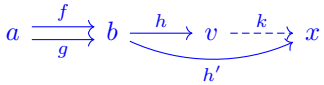


NOTE: In **Set**, coequalizers pick out the subset of a set b that is the intersection of the images $f : a \to b$ and $g : a \to b$.

Definition 12 (Coequalizer)**.** Let $a, b, v \in \mathcal{C}$ and let morphisms $f : a \to b$, $g : a \to b$, and $h : b \to v$ be given. The object v is a coequalizer if the following hold:

1. $h \circ f = h \circ g$.
2. For any other object $x \in \mathcal{C}$ with morphism $h' : b \to x$ such that $h' \circ f = h' \circ g$, there exists a unique morphism $k : v \to x$ such that $h' = k \circ h$.

This means that the following diagram commutes:

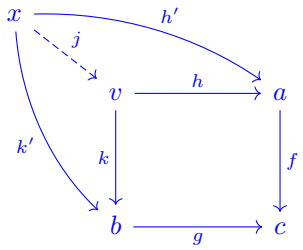


NOTE: In **Set**, pullbacks are the fiber product of two sets.

Definition 13 (Pullback)**.** Let $a, b, c, v \in \mathcal{C}$ and let morphisms $f : a \to c$, $g : b \to c$, $h : v \to a$, and $k : v \to b$ be given. The object v is a pullback if the following hold:

1. $f \circ h = g \circ k$.
2. For any other object $x \in \mathcal{C}$ with morphisms $h' : x \to a$, and $k' : x \to b$ such that $f' \circ h = g' \circ k$, there exists a unique morphism $j : x \to v$ such that $f' = f \circ j$ and $g' = g \circ j$.

This means that the following diagram commutes:

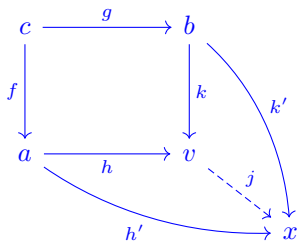


NOTE: In **Set**, pushouts are...

Definition 14 (Pushout)**.** Let $a, b, c, v \in \mathcal{C}$ and let morphisms $f : c \to a$, $g : c \to b$, $h : a \to v$, and $k : b \to v$ be given. The object v is a pushout if the following hold:

1. $h \circ f = k \circ g$.
2. For any other object $x \in \mathcal{C}$ with morphisms $h' : a \to x$, and $k' : b \to x$ such that $h' \circ f = k' \circ g$, there exists a unique morphism $j : v \to x$ such that $h' = j \circ h$ and $k' = j \circ k$.

This means that the following diagram commutes:



Cones and Limits

In this last part of the section we're going to discuss a generalization of all of the previous universal properties called cones and cocones. A cone consists of any finite collection of objects and morphisms between them, combined with an additional object v called the vertex of the cone and morphisms from v to all other objects in the collection. Such a cone is universal if for all other cones of that type in the category there is a unique morphisms from their vertex to v: i.e, $k : v' \to v$ exists and is unique.

A cocone consists of any finite collection of objects and morphisms between them, combined with an additional object v called the vertex of the cocone and morphisms from all other objects in the collection to v. Such a cocone is universal if for all other cocones of that type in the category there is a unique morphisms from v to their vertex: i.e, $k :\to v'$ exists and is unique.

Given any cone, if there is an instance of that cone for which all other instances have a unique morphism to it, that cone is the limit. Dually for cocones and colimits. In fact, it's possible to construct a category whose objects are the cones and whose morphisms are these unique factorizations. In this category, the limit is a terminal object. Dually for cocones and colimits again, except the colimit is an initial object in the constructed category of cocones instead of a terminal object.

Structure Between Categories

Kinds of Functors

Definition 15 (Full Functor)**.** A functor $F : \mathcal{C} \to \mathcal{D}$ is full if for each $x, y \in \mathcal{C}$, the map $\mathcal{C}(x, y) \to \mathcal{D}(Fx, Fy)$ is surjective.

Definition 16 (Faithful Functor)**.** A functor $F : \mathcal{C} \to \mathcal{D}$ is faithful if for each $x, y \in \mathcal{C}$, the map $\mathcal{C}(x, y) \to \mathcal{D}(Fx, Fy)$ is injective.

Definition 17 (Essentially Surjective On Objects)**.** A functor $F : \mathcal{C} \to \mathcal{D}$ is essentially surjective on objects if for each $y \in \mathcal{D}$, there is some $x \in \mathcal{C}$ such that y is isomorphic to Fc.

Definition 18 (Injective on Objects)**.** A functor $F : \mathcal{C} \to \mathcal{D}$ is injective on objects if for each $x \in \mathcal{C}$, there is a unique $y \in \mathcal{D}$ such that $y = Fx$.

NOTE: Play a huge role in category theory: i.e in the Yoneda lemma
NOTE: Representable functors require that the source category $\mathcal{C}$ be locally small so that the collections of morphisms between objects in the category are sets.

Definition 19 (Covariant Representable)**.** A functor $F : \mathcal{C} \to \mathbf{Set}$ is representable if there is an object $c \in \mathcal{C}$ such that the functor $\mathcal{C}(c, -) : \mathcal{C} \to \mathbf{Set}$ is naturally isomorphic to F.

Definition 20 (Contravariant Representable)**.** A functor $F : \mathcal{C}^{op} \to \mathbf{Set}$ is representable if there is an object $c \in \mathcal{C}$ such that the functor $\mathcal{C}(-, c) : \mathcal{C}^{op} \to \mathbf{Set}$ is naturally isomorphic to F.

Definition 21 (Forgetful)**.** A forgetful functor $F : \mathcal{C} \to \mathbf{Set}$ is a functor that sends objects in a category of structures-on-sets, like groups or preorders, to the underlying sets themselves. They "forget" the additional structure in the source category.

Definition 22 (Free)**.** A free functor $F : \mathbf{Set} \to \mathcal{C}$ is a functor that sends sets to the free construction on that set, like the free group on the set. They construct the desired structure on the set in a way that requires no choices.

Comparing Categories

Definition 23 (Subcategory)**.** A category $\mathcal{C}$ with a functor $F : \mathcal{C} \to \mathcal{D}$ that is both faithful and injective on objects is a subcategory of the domain $\mathcal{D}$. A functor F of this type is called an **embedding** of $\mathcal{C}$ into $\mathcal{D}$.

Definition 24 (Full Subcategory)**.** A category $\mathcal{C}$ with a functor $F : \mathcal{C} \to \mathcal{D}$ that is full, faithful, and injective on objects is a full subcategory of the domain $\mathcal{D}$. A functor F of this type is called a **full embedding** of $\mathcal{C}$ into $\mathcal{D}$.

Definition 25 (Natural Isomorphism)**.** A natural isomorphism is a natural transformation $\alpha : F \Rightarrow G$ where all of the c-components of the transformation $\alpha_c : Fc \to Gc$ are isomorphisms. This means there exists some other natural transformation $\eta : G \Rightarrow F$ such that $\eta_c \circ \alpha_c = id_{Fc}$ and $\alpha_c \circ \eta_c = id_{Gc}$.

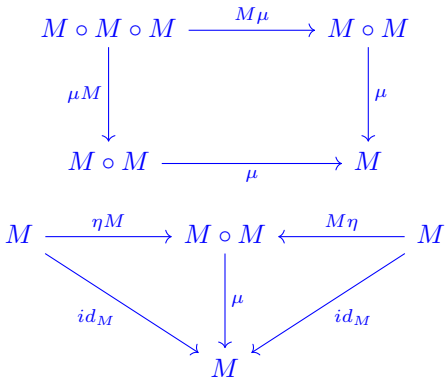
Definition 26 (Adjunctions)**.** An adjunction is a pair of functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ such that for any object $c \in \mathcal{C}$ and any object $d \in \mathcal{D}$ there is an isomorphism between the sets of functions between them and their images under the functors that is natural. In other words, that for every $c \in \mathcal{C}$ and $d \in \mathcal{D}$ there is a natural isomorphism $\mathcal{C}(c, Gd) \cong \mathcal{D}(Fc, d)$.

In this case, the functor F is the left adjoint functor and the functor G is the right adjoint functor.

Definition 27 (Monad)**.** Given a category $\mathcal{C}$, a monad on $\mathcal{C}$ is a functor $M : \mathcal{C} \to \mathcal{C}$ with the following two natural transformations:

1. $\eta : id_{\mathcal{C}} \Rightarrow M$ called the unit.
2. $\mu : M \circ M \Rightarrow M$ called the multiplication.

These natural transformation are required to make the following diagrams commute.



These commutative diagrams with the natural transformations η and μ create a kind of monoidal structure on the collection of endofunctors $M, M \circ M, M \circ M \circ M, etc..$

The Yoneda Lemma

Definition 28. The Yoneda lemma states that for any functor $F : \mathcal{C} \to \mathbf{Set}$ where $\mathcal{C}$ is locally small and any object $c \in \mathcal{C}$, there is a bijection between the collection of natural transformations going from the functor $\mathcal{C}(c, -)$ to F and the object $Fc \in \mathbf{Set}$.

Stated another way, this means that $Hom(\mathcal{C}(c, -), F) \cong Fc$ for any functor $F : \mathcal{C} \to \mathbf{Set}$ and all objects $c \in \mathcal{C}$.

Adding Structure to Categories

Definition 29 (Monoidal Categories)**.** A category $\mathcal{C}$ is monoidal if the following exist.

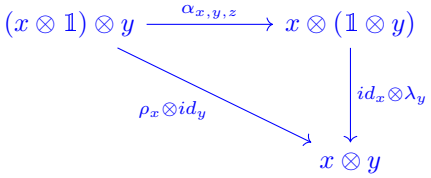
1. A functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ called the monoidal product.
2. An object $\mathbb{1} \in \mathcal{C}$ called the unit.

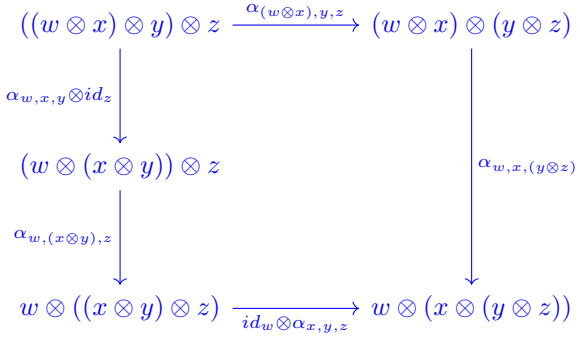
3. A natural isomorphism $\alpha : (a \otimes b) \otimes c \Rightarrow a \otimes (b \otimes c)$ called the associator, with components $\alpha_{x,y,z} : (x \otimes y) \otimes z \to x \otimes (y \otimes z)$.

4. A natural isomorphism $\lambda : \mathbb{1} \otimes a \Rightarrow a$ called the left unitor with components $\lambda_x : \mathbb{1} \otimes x \to x$.

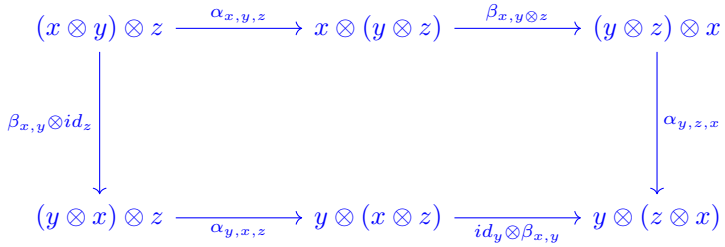
5. A natural isomorphism $\rho : a \otimes \mathbb{1} \Rightarrow a$ called the right unitor with components $\lambda_x : x \otimes \mathbb{1} \to x$.

All of the above must exist such that the following two diagrams, called the triangle identity and the pentagon identity, commute.





Definition 30 (Symmetric Monoidal Category)**.** A monoidal category equipped with a natural isomorphism $\beta : a \otimes b \Rightarrow b \otimes a$, called its braiding, is symmetric if $\beta_{y,x} \circ \beta_{x,y} = id_{x \otimes y}$ for all $x, y \in \mathcal{C}$ and if the following diagram, called the hexagon identity, commutes.



Definition 31 (Cartesian Monoidal Category)**.** A monoidal category $\mathcal{C}$ is cartesian when its monoidal product is its categorical product and its monoidal unit is its terminal object. This obviously means that $\mathcal{C}$ must contain all products and have a terminal object.

An example of this would be **Set**, where the Cartesian product is the monoidal product and the singleton set is the monoidal unit.

Categorical System Theory

Systems are everywhere in science and engineering. Whether discrete or continuous, deterministic or stochastic, all such systems have two things in common: states that they can be in, and rules for how the system's current state changes. These two features alone describe what are called closed systems. Closed systems are systems that don't interact with others or with an outside environment that they're a part of. This is a hobbling limitation. In order to open these systems up we need to give them an interface. We need them to be able to accept inputs that shape the way they evolve, and we need them to be able to expose some part of their current state to their surrounding environment. This is the gift that category theory will give us. By opening our systems up, we open them to the power of composition. We can connect them.

The Category $\mathbf{Lens}_{\mathcal{C}}$

Given a cartesian category, it's possible to construct a new category of systems whose states are drawn from the objects of your base category and whose rules for updating their state are drawn from the morphism of your base category. We call this category $\mathbf{Lens}_{\mathcal{C}}$, where $\mathcal{C}$ is the base category. The objects in this category are called arenas and the morphisms between them are called lenses.

Definition 32 (Lenses)**.** Given a cartesian category $\mathcal{C}$ and objects $A^-, A^+, B^-, B^+ \in \mathcal{C}$, a lens consists of a passforward map $f : A^+ \to B^+$ and a passback map $f^\# : A^+ \times B^- \to A^-$ between two arenas as follows:

$$\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$$

For our purposes we'll be sticking to two such categories: the category of lenses over the category of sets and functions, $\mathbf{Lens}_{\mathbf{Set}}$, for discrete systems and the category of Euclidean spaces and smooth functions, $\mathbf{Lens}_{\mathbf{Euc}}$, for differential systems.

Definition 33 (The Category $\mathbf{Lens}_{\mathcal{C}}$)**.** Given the cartesian category $\mathcal{C}$, the category $\mathbf{Lens}_{\mathcal{C}}$ has the following properties.

1. A collection of objects called arenas. An arena $\begin{pmatrix} A^- \\ A^+ \end{pmatrix}$ is a pair of objects in $\mathcal{C}$.
2. For each pair of arenas a collection of morphisms $\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$ called lenses.

3. For each arena an identity lens $\begin{pmatrix} \pi_2 \\ id_{A^+} \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} A^- \\ A^+ \end{pmatrix}$ where the passback map π_2 is the projection $\pi_2 : A^+ \times A^- \to A^-$.

4. For any two compatible lenses a composite lens as follows:

$$\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$$

$$\begin{pmatrix} g^\# \\ g \end{pmatrix} : \begin{pmatrix} B^- \\ B^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} C^- \\ C^+ \end{pmatrix}$$

$$\begin{pmatrix} g^\# \\ g \end{pmatrix} \circ \begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} C^- \\ C^+ \end{pmatrix}$$

   such that the passforward map is defined as $a^+ \mapsto g(f(a^+))$, and the passback map is defined as $(a^+, c^-) \mapsto f^\#(a^+, g^\#(f(a^+), c^-))$.

Lens$_\mathcal{C}$ is also a monoidal category with the monoidal product being defined as follows.

1. The monoidal unit is the arena $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

2. Given lenses $\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$ and $\begin{pmatrix} g^\# \\ g \end{pmatrix} : \begin{pmatrix} C^- \\ C^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} D^- \\ D^+ \end{pmatrix}$, the monoidal product

$$\begin{pmatrix} f^\# \\ f \end{pmatrix} \otimes \begin{pmatrix} g^\# \\ g \end{pmatrix} : \begin{pmatrix} A^- \times C^- \\ A^+ \times C^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \times D^- \\ B^+ \times D^+ \end{pmatrix}$$

   such that the passforward map is defined as $(a^+, c^+) \mapsto (f(a^+), g(c^+))$ and the passback map is defined as $((a^+, c^+), (b^-, d^-)) \mapsto ((f^\#(a^+, b^-)), g^\#(c^+, d^-))$.

References

[Che22] Eugenia Cheng. *The joy of abstraction: an exploration of math, category theory, and life*. Cambridge University Press, 2022.

[FS19] Brendan Fong and David I Spivak. *An invitation to applied category theory: seven sketches in compositionality*. Cambridge University Press, 2019.

[Gol14] Robert Goldblatt. *Topoi: the categorial analysis of logic*. Elsevier, 2014.

[Lei14] Tom Leinster. *Basic category theory*, volume 143. Cambridge University Press, 2014.

[LR03] F William Lawvere and Robert Rosebrugh. *Sets for mathematics*. Cambridge University Press, 2003.

[Mil18] Bartosz Milewski. *Category theory for programmers*. Blurb, 2018.

[ML13] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[Mye22] David Jaz Myers. Categorical systems theory. *Manuscript in preparation*, 2022.

[nLa24] nLab authors. symmetric monoidal category. `https://ncatlab.org/nlab/show/symmetric+monoidal+category`, February 2024. Revision 54.

[Rie17] Emily Riehl. *Category theory in context*. Courier Dover Publications, 2017.

[Ros22] Daniel Rosiak. *Sheaf theory through examples*. MIT Press, 2022.