2. Categories, Functors, and Natural Transformations

We'll jump straight into the definition.

Definition 2.1. A *category* $\mathcal{C}$ consists of the data

- a collection $|\mathcal{C}|$ of *objects* $X, Y, \dots$
- a set $\mathcal{C}(X, Y)$, called the *hom set*, of *arrows* or *morphisms* $f, g, \dots : X \to Y$
- for any pair $f \in \mathcal{C}(X, Y), g \in \mathcal{C}(Y, Z)$, a *composition* $g \circ f = f /\!\!/ g \in \mathcal{C}(X, Z)$.
- for any $X \in |\mathcal{C}|$, a specified *identity* arrow $\mathbb{1}_X \in \mathcal{C}(X, X)$

satisfying the conditions

- $(f \circ g) \circ h = f \circ (g \circ h)$ when these make sense (associativity)
- $f \circ \mathbb{1}_X = f = \mathbb{1}_Y \circ f$ for $f : X \to Y$ (unitality)

Remark 2.2. Note that we have a *collection* of objects and not a *set*—this is to avoid Russel's paradox (the set of all sets that don't contain themselves) style paradoxes. When the collection is an actual set, we say that the category is *small.* Rest assured that mathematicians have provided this situation a logically valid formal account, but this discussion is outside the scope of our course.

For $f \in \mathcal{C}(X, Y)$, we respectively call X and Y the *domain* and *codomain* of f. We say an arrow is an *endomorphism* when its domain and codomain are equal. We say a sequence of arrows $f_i : X_i \to X'_i$ is *composable* if for all i, $X_{i+1} = X'_i$. When the category is understood from context, or when we wish to speak about generic cases, we will write $\hom(X, Y)$ for $\mathcal{C}(X, Y)$. A *subcategory* $\mathcal{D}$ of $\mathcal{C}$ consists of a subcollection of $\mathcal{C}$ arrows (along with their corresponding domain and codomain) such that they still assemble into a category. Before presenting examples of categories, we warn that these will come in two distinct flavors, which one can conceive as "categories as objects" and "categories of objects." The former concerns itself with reinterpreting simpler objects as special cases of categories and the latter concerns itself with assembling the entire theory of a simpler object into a single category. We begin with instances of the former.

Example 2.3. Objects we have seen in the prior lectures can be seen as categories.

(1) any set S can be reinterpreted as a category $\overline{S}$, whose objects are the elements of S, and whose morphisms only consist of identity arrows. More formally, $\overline{S}$ can be characterized as follows.
  - $|\overline{S}| = S$
  - $\overline{S}(x, y) = \begin{cases} \{\mathbb{1}_x\} & x = y \\ \varnothing & x \neq y \end{cases}$

  We call a category $\mathcal{C}$ *discrete* if it only contains identity arrows. We can hence identify discrete categories with sets.

(2) any preorder $(P, \preceq)$ can be seen as a category $\overline{P}$ whose objects are again elements of P, but whose arrows $x \to y$ correspond precisely to statements $x \preceq y$, as was visualized in our Hasse diagrams. More formally, $\overline{P}$ can be characterized as follows.
  - $|\overline{P}| = P$
  - $\overline{P}(x, y) = \begin{cases} \star & x \preceq y \\ \varnothing & x \not\preceq y \end{cases}$

  This perhaps requires some explanation. Recall that $\star$ is a generic singleton. You may be concerned that we did not label the element contained inside the singleton. The point is that this is not necessary—the singleton merely signals that there *exists* an arrow $x \to y$. Since it is either true or false that $x \preceq y$, there is no need to label this arrow. Another way to look at this is to think of the homs $\overline{P}(X, Y)$ not as sets but as *truth values*, i.e. elements of $\mathbb{B}$, where $\star$ corresponds to $\top$ and $\varnothing$ to $\bot$. This makes $\overline{P}(-,-) : P \times P \to \mathbb{B}$, which aligns entirely with the fact that the type of $\preceq$ was $P \times P \to \mathbb{B}$. Furthermore, note that the existence of an identity arrow forces $\overline{P}(x, x)$ to be non-empty, thus instantiating reflexivity. In addition, the existence of a composition $x \to z$ for any pair of arrows $x \to y, y \to z$ implies transitivity. We call a category $\mathcal{C}$ *thin* when $\mathcal{C}(X, Y)$ contains at most one element. We can hence identify thin categories with preorders.

(3) any monoid $(M, \cdot, e)$ can be seen as a category $\mathbf{B}M$ which, in contrast to the prior examples, *has only one object*, say $\bullet$—the name does not matter!—and whose arrows, which are just the endomorphisms $\mathbf{B}M(\bullet, \bullet)$, are precisely the elements of M. Composition of arrows is then given by our binary operation and the identity arrow by our identity element. More formally, $\mathbf{B}M$ can be characterized as follows.
   - $|\mathbf{B}M| = \{\bullet\}$
   - $\mathbf{B}M(\bullet, \bullet) = M$
   - $f \circ g = f \cdot g$
   - $\mathbb{1}_\bullet = e$

   In fact, any *one-object* category $\mathcal{C}$ can be identified with a monoid.

Any subset $T \subseteq S$ yields a subcategory $\mathcal{D}T$ of $\mathcal{D}S$. The analogous is true for suborders of a preorder, such as $\langle n \rangle$ of $\langle n \cdot m \rangle$, or submonoids of a monoid, such as $(\mathbb{N}, +, 0)$ of $(\mathbb{Z}, +, 0)$.

In addition to these examples, we can depict any finite category (those with a finite amount of arrows and hence objects), via a *quiver*, which is merely a generalization of directed graph that allows for multiple edges between any two vertices. This includes as a special case any Hasse diagram of a poset. A non-thin example would be categories such as $\bullet \rightrightarrows \ast$.

We now list examples of our second flavor of category, which involve assembling all objects of a certain type along with appropriate choices of arrows for that type.

Example 2.4. The following examples involve gathering into a category the theory of previously encountered objects.

(1) The category **Set** has objects sets $X, Y, \dots$ and arrows $\mathbf{Set}(X, Y)$ maps $f, g \dots : X \to Y$.
(2) The categories **Pre**, **Pos**, **Tot** respectively have objects preorders, posets, and total orders; and arrows $\hom((P, \preceq), (Q, \sqsubseteq))$ monotonic maps.
(3) The categories **Mon** and **Com** respectively have objects monoids and commutative monoids; both have arrows $\mathbf{Mon}((M, \cdot, e_M), (N, *, e_N))$ *monoid homomorphisms*, i.e. maps $f : M \to N$ with $f(x \cdot y) = fx * fy$, $f(e_M) = e_N$.
(4) The category $\mathbf{Vect}_k$ has objects vector spaces $V, W, \dots$ over the field k and arrows $\mathbf{Vect}_k(V, W)$ k-linear maps.
(5) The category **Rel** has objects sets and arrows $\mathbf{Rel}(X, Y)$ maps $X \times Y \to \mathbb{B}$.

One can conceive of **Pos** as a subcategory of **Pre**, **Tot** as a subcategory of **Pos**, **Com** as a subcategory of **Mon**, and, as we shall discuss further, **Set** as a subcategory of **Rel**.

We say an arrow $f : X \to Y$ is an *isomorphism* when there exists $g : Y \to X$ such that $f \circ g = \mathbb{1}_Y$ and $g \circ f = \mathbb{1}_X$. We then write $f : X \xrightarrow{\cong} Y$ or just $X \cong Y$, and say that X and Y are *isomorphic*. A category for which *all* arrows are isomorphisms is called a *groupoid*. An isomorphism endomorphism is also called an *automorphism*. A one-object groupoid, or, equivalently, a monoid for which all elements are invertible, is called a *group*. Any group can be seen as the collection of automorphisms—often conceived of as self-symmetries—of some object. The category **Grp** of groups is a subcategory of **Mon**. In the context of **Set**, an isomorphism is a bijection. We have already seen isomorphisms of preorders in the previous assignment. In the context of *a* preorder $(P, \preceq)$, seen as a category $\overline{P}$, an isomorphism $x \cong y$ is the case when $x \preceq y$ and $y \preceq x$. A poset can then be defined as a preorder for which an isomorphism is an equality. In fact, all of our theorems regarding the uniqueness of the likes of upper sets and universal objects in posets, can be immediately generalized to preorders by simply adding the phrase "up to isomorphism" at the end of our statements; e.g. "the join $x \vee y$ is unique up to isomorphism" simply means that, if z, z' both satisfy the condition for being a join, then it must be that $z \cong z'$. In category theory, we usually like to work in contexts in which isomorphismic objects are *not* identified as equal, even though they essentially "behave the same way." In fact, may constructions become so ill behaved if we make such an identification that we call this situation *evil*.

What can we do with categories aside from keep track of a wildly varying class of mathematical objects? Just as maps are important for studying sets, we need a notion of arrow for category. In fact, this can be expressed in a satisfyingly self similar manner: there is a category **Cat** whose objects are (small) categories and arrows $\mathcal{C} \to \mathcal{D}$ are so called *functors*, which we now define formally.

Definition 2.5. For categories $\mathcal{C}, \mathcal{D}$, a *functor* $F : \mathcal{C} \to \mathcal{D}$ consists of the data

- a map $|F| : |\mathcal{C}| \to |\mathcal{D}|$ between the categories' objects
- a map $F_{X,Y} : \mathcal{C}(X, Y) \to \mathcal{D}(|F|X, |F|Y)$ for all hom sets $\mathcal{C}(X, Y)$

satisfying the following conditions

- $F_{X,Z}(f \circ g) = F_{Y,Z}f \circ F_{X,Y}g$ (preservation of composition)
- $F_{X,X}(\mathbb{1}_X) = \mathbb{1}_{|F|X}$ (preservation of identity)

One somewhat trivial example of a functor is the *identity functor* $\mathbb{1}_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ for any category $\mathcal{C}$ whose behavior is given by $|\mathbb{1}_{\mathcal{C}}| = \mathbb{1}_{|\mathcal{C}|}$ and $[\mathbb{1}_{\mathcal{C}}]_{X,Y} = \mathbb{1}_{\mathcal{C}(X,Y)}$. There are in fact more interesting examples of functors that don't involve "altering data." Let's consider a lower level example of what we mean. There is a so called *inclusion* map $i : \mathbb{Z} \to \mathbb{Q} :: n \mapsto n$, which does not alter the value of its argument n. This map does however alter the *type* of n and hence instantiates in the form of a map an act of reconceptualization of an integer as a rational. The same occurs at the level of categories, and we have in fact seen several instances of such in Example 2.3.

Example 2.6. The following functors instantiate the reconceptualization of simpler objects as categories.

(1) The *discrete* functor $\mathbf{D} : \mathbf{Set} \to \mathbf{Cat}$ is given by
   - $|\mathbf{D}|S = \overline{S}$
   - $\mathbf{D}_{S,T}f = \overline{f}$, where $\overline{f} : \overline{S} \to \overline{T}$ is the functor given by
     - $|\overline{f}| : |\overline{S}| \to |\overline{T}| = f : S \to T$

- $\overline{f}_{x,x}\mathbb{1}_x = \mathbb{1}_{fx}$

  In plain English, a functor between discrete categories is just a map between the object sets.

(2) The *thin* functor $\mathbf{T} : \mathbf{Pre} \to \mathbf{Cat}$ is given by
- $|\mathbf{T}|(P, \preceq) = \overline{P}$
- $\mathbf{T}_{(P,\preceq),(Q,\sqsubseteq)} f = \overline{f}$, where $\overline{f} : \overline{P} \to \overline{Q}$ is the functor given by
  - $\overline{f} : \overline{P} \to \overline{Q} = f : (P, \preceq) \to (Q, \sqsubseteq)$
  - $\overline{f}\star = \star$

  In plain English, a functor between thin categories is just a monotonic map between the corresponding preorders.

(3) The *delooping* functor $\mathbf{B} : \mathbf{Mon} \to \mathbf{Cat}$ is the functor given by
- $|\mathbf{B}|(M, \cdot, e_M) = \mathbf{B}M$
- $\mathbf{B}_{(M,\cdot,e_M),(N,*,e_N)} f = \mathbf{B}f$, where $\mathbf{B}f$ is the functor given by
  - $|\mathbf{B}f|\bullet = \bullet$
  - $[\mathbf{B}f]_{\bullet,\bullet} : \mathbf{B}M(\bullet, \bullet) \to \mathbf{B}N(\bullet, \bullet) = f : M \to N$.

  In plain English, a functor between one-object categories is just a monoid homomorphism between their respective set of endomorphism.

For the sake of removing clutter, we will henceforth drop the adornments on F that distinguish its behavior on objects and arrows, and simply write $FX, FY, \ldots$ and $Ff, Fg, \ldots$ for the result of applying F to either objects or arrows.

There are other inclusion functors worth mentioning. Firstly there is the sequence of inclusions $\mathbf{Pre} \to \mathbf{Pos} \to \mathbf{Tot}$ defined in the obvious way. There is an inclusion $\mathbf{Set} \to \mathbf{Pos}$ which is also discrete in the sense that the only relations are those required by reflexivity. A slightly more interesting example is the *graph* functor $G : \mathbf{Set} \to \mathbf{Rel}$. This functor is the identity on objects, and takes a map $f : X \to Y$ to the relation $\tilde{f}$ given by

$$\tilde{f}(x, y) = \begin{cases} \top & fx = y \\ \bot & fx \neq y \end{cases}$$

Said otherwise, since a relation $X \times Y \to \mathbb{B}$ is, by Proposition **??**, equivalent to a subset of $X \times Y$ corresponding to the points sent to true. In this case, this is precisely the set of points $\{(x, fx)\} \subseteq X \times Y$, which is the familiar notion of a graph of a function, most famously in the case of drawing curves in the 2 dimensional plane $\mathbb{R}^2$ as a way to represent maps $\mathbb{R} \to \mathbb{R}$.

Another class of functors are the so called *forgetful* functors, which take an object of a certain type and simply "forget" that it comes equipped with certain structures or properties. These often come in the form of *underlying set* functors, which take an object consisting of a set equipped with other features and merely returns the set. There are such functors U from $\mathbf{Cat}, \mathbf{Pre}, \mathbf{Mon}, \mathbf{Vect}_k, \ldots$ to $\mathbf{Set}$. Sometimes one needn't forget the entirety of an object's structure, e.g. in the forgetful functor $\mathbf{Vect}_k \to \mathbf{Mon}$ that takes a vector space V to the monoid $(V, +, 0)$, forgetting the structure of the scalar multiplication, along with nice properties, such as commutativity and the existence of inverses, that $+$ satisfies. Although the structures are now gone, note that it is not the case that these properties are magically no longer satisfied; rather, the codomain category to which the object is sent merely no longer requires that property for an object to inhabit it. An example that involves no loss of structure—only a property—is the forgetful functor $\mathbf{Com} \to \mathbf{Mon}$ that does nothing to the argument aside from situating it in a context which no longer requires commutativity. Later in the section, we will give a formal definition for what it even means for something to be a structure or a property.

Dually, there is a class of functors called *free functors*, that take an object that lacks a certain structure and equip it "freely" with that structure, in the sense of giving it the simplest "least arbitrary" version of the desired new structure as possible. We will return to what this means formally in a future section. The two discrete functors from **Set** to **Pre** and **Cat** are both turn out to be examples of this. We now introduce two new examples that have a more playful feel.

Example 2.7. Given a set S, how do we use S to construct a k-vector space? Recall that a k-vector space is a set closed under k-linear combination, i.e. $u, u' \in V \Rightarrow cu+c'u' \in V$ for all $c, c' \in k$. Naturally the set $kS = \{\sum_{x\in S} c_x x \mid c_x \in k\}$ of all linear combinations is in S is by construction closed under linear combinations and henc could be conceived of as a vector space. I.e. we map a set S to a vector space whose basis is S. Recall that a linear map $L : V \to W$ is determined by its behavior on the basis. Thus we can send a map $f : S \to T$ to the linear map $kf : kS \to kT$ given by linearly extending the behavior of f on S. The reader is encouraged to check that this indeed preserves compositions and identities. Therefore this procedure assembles into the free functor $F : \mathbf{Set} \to \mathbf{Vect}_k$.

Remark 2.8. Note that, although the free functor resembles in content the span of a set of vectors, it is slightly different. The span maps a set $S \subset V$ of vectors, already pre-specified as being in a given vector space V, to the set of their linear combinations, which, in the case where the vectors are not linearly independent, yields a subspace of V whose basis is smaller than S. In turn, the free vector space takes arguments in arbitrary sets, for whom there do not yet exist any predefined linear relations among their elements. This is analogous to the distinction between union, which takes as argument subsets of a prespecified ambient set, and the disjoint union which takes as argument any two sets, for whom there do not yet exist any predefined equality relations between their elements.

A rather similar procedure can be considered for monoids.

Example 2.9. Given a set S, how do we use S to construct a monoid? A monoid can be conceived of as a set closed under n-ary versions, including the case where $n = 0$ of a specified binary operation. In the same vein as the above example, the set $S^* = \{s_0 s_1 \cdots s_{n-1} \mid s_i \in S\}$, i.e. it can be seen as the set of *words* in the *alphabet* S. Then the concatenation of words $(s_0 \cdots s_{m-1}, s_m \cdots s_n) \mapsto s_0 \cdots s_{n-1} s_n \cdots s_m$ yields the monoid's operation, with the empty word serving as its unit. We can extend a map $f : S \to T$ to a map $f^* : S^* \to T^* :: s_0 \cdots s_{n-1} \mapsto f(s_0) \cdots f(s_{n-1})$. This procedure automatically preserves identities and compositions, and thus assembles into the free functor $F : \mathbf{Set} \to \mathbf{Mon}$.

A highly related functor is the *list* endofunctor $\mathrm{List} : \mathbf{Set} \to \mathbf{Set}$, which maps a set S to the set $\mathrm{List}\, S$ consists of all the lists $[s_0, s_1, \ldots, s_{n-1}]$ with entries in S. We extend a map $f : S \to T$ to a map $\mathrm{List}\, f : \mathrm{List}\, S \to \mathrm{List}\, T$ by applying f to each individual entry. Note that lists seem to only differ in notation, via their use of brackets and commas, from the words given by the free monoid functor. Another difference is the fact that the free monoid FS is a monoid, whereas $\mathrm{List}\, S$ is a set. We could, however, consider the following composition with the forgetful functor.

$$\mathbf{Set} \xrightarrow{F} \mathbf{Mon} \xrightarrow{U} \mathbf{Set}$$

It turns out that, when we apply this composite functor $UF : \mathbf{Set} \to \mathbf{Set}$, to a set S, we see that there is an isomorphism, i.e. a bijection, $UF(S) \to \mathrm{List}\, S$ given by

$$s_0 \cdots s_{n-1} \mapsto [s_0, \ldots, s_{n-1}]$$

This isomorphism not only holds for any set S but it ends up instantiating an *isomorphism of functors* $UF \cong \mathrm{List}$. We will define what this means, upon witnessing more examples, later in the section.

We have actually already encountered a trio of functors corresponding to the powerset. For example, there is a functor $\mathcal{P} : \mathbf{Set} \to \mathbf{Pos}$ mapping a set to its powerset $S \mapsto (\mathcal{P}S, \subseteq)$ and a map to its direct image $[f : S \to T] \mapsto [f_* : \mathcal{P}S \to \mathcal{P}T]$. Alternatively, there is another powerset functor that takes a map to its indirect image. We can also choose to forget the ordering and consider the functor $U\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$, which is often denoted as just $\mathcal{P}$. What about the preimage? Recall that for a map $f : S \to T$, the preimage is a map $f^* : \mathcal{P}T \to \mathcal{P}S$ going the other way! We can consider this as a functor via the *opposite category* construction, which generalizes the opposite order construction of the prior section. More precisely, given a category $\mathcal{C}$, we define its opposite category $\mathcal{C}^{\mathrm{op}}$ as having the same objects but for which arrows are just formally reversed, i.e. $\mathcal{C}^{\mathrm{op}}(X, Y) = \mathcal{C}(Y, X)$. We sometimes call a functor $\mathcal{C}^{\mathrm{op}} \to \mathcal{D}$ a *contravariant functor* $\mathcal{C} \to \mathcal{D}$. To contrast with contravariant, we call a functor that doesn't reverse arrows *covariant*. We can then define the contravariant powerset functor $\mathcal{P} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Pos}$ given by $S \mapsto (\mathcal{P}, \subseteq)$ and $[f : S \to T] \mapsto [f^* : \mathcal{P}T \to \mathcal{P}S]$.

There is one particularly important class of functors called *hom functors*. Given a category $\mathcal{C}$ and an object $X \in |\mathcal{C}|$, we define the *covariant hom functor*

$$\mathcal{C}(X, -) : \mathcal{C} \to \mathbf{Set}$$

via the behavior

- $Y \mapsto \mathcal{C}(X, Y)$
- $[f : Y \to Y'] \mapsto \lambda\varphi.[\varphi \mathbin{/\!\!/} f] : \mathcal{C}(X, Y) \to \mathcal{C}(X, Y')$.

Unpacking the behavior on maps, we want the functor $\mathcal{C}(X, -)$ to map an arrow $f : X \to Y$ to a map $\mathcal{C}(X, Y) \to \mathcal{C}(X, Y')$, i.e. a procedure for taking an arrow $\varphi : X \to Y$ and producing an arrow of type $X \to Y'$. We do this by simply taking the composition:

$$X \xrightarrow{\varphi} Y \xrightarrow{f} Y'.$$

We choose the diagrammatic composition notation $/\!\!/$ in place of $\circ$ to make the composition order more intuitive. This is a functor since $f \mathbin{/\!\!/} g$ gets mapped to $\lambda\varphi.[\varphi \mathbin{/\!\!/} f \mathbin{/\!\!/} g]$. Dually, we have a *contravariant hom functor*

$$\mathcal{C}(-, X) : \mathcal{C} \to \mathbf{Set}$$

via the behavior

- $Y \mapsto \mathcal{C}(Y, X)$
- $[f : Y \to Y'] \mapsto \lambda\varphi.[f \mathbin{/\!\!/} \varphi] : \mathcal{C}(Y', X) \to \mathcal{C}(Y, X)$.

Unpacking the behavior on maps, we want the functor $\mathcal{C}(-, X)$ to map an arrow $f : X \to Y$ to a map $\mathcal{C}(Y', X) \to \mathcal{C}(Y, X)$, i.e. a procedure for taking an arrow $\varphi : Y' \to X$ and producing an arrow of type $Y \to X$. We do this by simply taking the composition:

$$Y \xrightarrow{f} Y' \xrightarrow{\varphi} X.$$

Since this functor is contravariant, $f \mathbin{/\!\!/} g$ gets mapped to $\lambda\varphi.[g \mathbin{/\!\!/} f \mathbin{/\!\!/} \varphi]$. These functors will play a central role in the subsequent section.

Another interesting family of functors is given by binary operations on objects, such as the Cartesian Product, taking a pair of sets X, Y to $X \times Y$. To define these as functors, we actually need the notion of a *product of categories*. Given two categories $\mathcal{C}, \mathcal{D}$, we define the category $\mathcal{C} \times \mathcal{D}$ as having objects pairs (X, Y), for $X \in |\mathcal{C}|$ and $Y \in |\mathcal{D}|$ and arrows $(X, Y) \to (X', Y')$ as simply pairs of arrows $f : X \to X'$ and $g : Y \to Y'$. More formally:

- $|\mathcal{C} \times \mathcal{D}| = |\mathcal{C}| \times |\mathcal{D}|$
- $[\mathcal{C} \times \mathcal{D}]((X, Y), (X', Y')) = \mathcal{C}(X, X') \times \mathcal{D}(Y, Y')$.

Then the Cartesian product can be seen as a functor **Set** $\times$ **Set** $\to$ **Set** mapping (X, Y) to $X \times Y$ and an arrow $(f : X \to X', g : Y \to Y')$ to the map

$$X \times Y \to X' \times Y' :: (x, y) \mapsto (fx, gy).$$

In a similar vein, we can define the disjoint union as a functor of the same type, or, analogously, the direct sum of vector spaces as a functor **Vect** $\times$ **Vect** $\to$ **Vect**. These functors will also play a crucial role in the next section.

Our final example of functor is the notion of *diagram*. A *diagram of shape* Δ *in* $\mathcal{C}$ is merely a functor $\Delta \to \mathcal{C}$, where Δ is often finite and is represented graphically as a quiver. For example, a diagram F of the shape $a \xrightarrow{f} b$ in **Set** is merely a map $Fa \xrightarrow{Ff} Fb$; and more generally a diagram of this shape in $\mathcal{C}$ is just a $\mathcal{C}$-arrow. A diagram F in $\mathcal{C}$ of the shape

$$\begin{tikzcd} a \arrow[r, "f"] & b \\ b \arrow[u, "g"] \arrow[ur, "gf"'] & \end{tikzcd}$$

is a *commutative* triangle in $\mathcal{C}$:

$$\begin{tikzcd} Fa \arrow[r, "Ff"] & Fb \\ b \arrow[u, "Fg"] \arrow[ur, "F(gf)"'] & \end{tikzcd}$$

By a *commutative diagram* we mean a diagram for which any two composition paths between any two objects give the same map. We can formally encode the notion of a commutative diagram as a diagram of shape Δ where Δ is a preorder. This is because, since preorders are thin categories, i.e. have at most one arrow between any two objects, that it must be the case that any composition path gives precisely this unique object. Because functors preserve all compositions, they also preserve all commutative diagrams. One particularly noteworthy diagram shape is the so called *commutative square*:

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[d, "g"'] & b \arrow[d, "g'"] \\ a' \arrow[r, "f'"'] & b' \end{tikzcd}$$

Just as a set map can be better understood by considering whether or not it is injective or surjective, an analogous, yet more granular, vocabulary exists for a functor. In particular, we say the functor $F : \mathcal{C} \to \mathcal{D}$ is

- *faithful* if $F_{X,Y} : \mathcal{C}(X,Y) \to \mathcal{D}(X,Y)$ is always injective
- *full* if $F_{X,Y} : \mathcal{C}(X,Y) \to \mathcal{D}(X,Y)$ is always surjective
- *essentially wide* if for all $Z \in |\mathcal{D}|$, there is some $X \in |\mathcal{C}|$ for which $FX \cong Z$.

The final condition is also called *essentially surjective on objects.* We drop the "essentially" from both names if the isomorphism is replaced by an equality. The up-to-isomorphism notion however will play a more fundamental role. When F is both faithful and full, we say F is *fully faithful.* These functors satisfy $FX \cong FY \Rightarrow X \cong Y$, since any pair of $\mathcal{D}$ arrows $\alpha : FX \to FY, \beta : FY \to FX$ with $\alpha \circ \beta = \mathbb{1}_{FY}, \beta \circ \alpha = \mathbb{1}_{FX}$ will have unique preimages $F^{-1}\alpha : X \to Y, F^{-1}\beta : Y \to X$ which, by functoriality, will satisfy $F^{-1}\alpha \circ F^{-1}\beta = \mathbb{1}_Y, F^{-1}\beta \circ F^{-1}\alpha = \mathbb{1}_X$.