Lecture 8: Functors, Natural Transformations

Lecturer: Max S. New
Scribe: Wen Plotnick

February 6, 2023

1 Definition and examples of functors

Previously, we have seen how we make categories from mathematical objects and the structure preserving morphisms between them. A natural question is what the analogous notion for structure preserving maps between categories would be, like we do for lots of mathematical objects. The correct notion ends up being the following.

Definition 1 (Functor)**.** *Suppose $\mathcal{C}$ and $\mathcal{D}$ are categories. A functor $F : \mathcal{C} \to \mathcal{D}$ is an object containing the following information:*

1. *An assignment of objects $F_0 : \mathcal{C}_0 \to \mathcal{D}_0$ which maps every object in $\mathcal{C}_0$ to an object in $\mathcal{D}_0$,*
2. *For all $a, b \in \mathcal{C}_0$, an assignment of morphisms $F_1^{a,b} : \mathcal{C}_1(a, b) \to \mathcal{D}_1(F_0(a), F_0(b))$,*
3. *such that F_1 preserves the identity morphism, i.e. for all objects $a \in \mathcal{C}_0$*

$$F_1^{a,a}(\mathrm{Id}_a) = \mathrm{Id}_{F_0(a)},$$

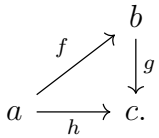
4. *and such that F_1 preserves composition, i.e. for all objects $a, b, c \in \mathcal{C}_0$ and morphisms $f \in \mathcal{C}_1(b, c)$, $g \in \mathcal{C}_1(a, b)$, we have*

$$F_1^{a,c}(f \circ g) = F_1^{b,c}(f) \circ F_1^{a,b}(g).$$

To understand where this definition comes from, recall we can think of categories as a sort of generalization of preorders where we have some notion of equality for our order relations. Then (2) in the above definition becomes a generalization of the condition of being monotone. To explain, if $\mathcal{C}$ and $\mathcal{D}$ were both thin categories, i.e. just preorders by last lecture, suppose we have $a, b \in \mathcal{C}_0$ such that $a \leq b$. Our morphisms $F_1^{a,b} : \mathcal{C}_1(a, b) \to \mathcal{D}_1(F_0(a), F_0(b))$ implies there is some morphism $F_1^{a,b}(a \leq b) \in \mathcal{D}_1(F_0(a), F_0(b))$, which implies $F_0(a) \leq F_0(b)$. So, our functor F is monotone when reinterpreted as a function between the underlying preorders. As we are working in general categories, not just thin categories, however, we also have a

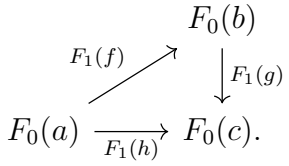
notion of equality for our morphisms. Therefore, we also add conditions (3) and (4) to preserve the structure of our equality relations for morphisms.

Another way to think about functor $F : \mathcal{C} \to \mathcal{D}$ is that it gives "something shaped like $\mathcal{C}$" inside $\mathcal{D}$. To make this make sense, in category theory we often draw diagrams such as the following:



We interpret the arrows as morphisms in a category, and the objects they point between as the source and target of the morphism. So the above tells us we have morphisms $f : a \to b$, $g : b \to c$, and $h : a \to c$. Further, we say the diagram is commutative when every path through the diagram with the same endpoints, taking the composition of morphisms as we go, yields equal morphisms. So if the above is a commutative diagram, we know $g \circ f = h$, as we can go from a to c either directly through h, or indirectly through b via f and g.

Returning to the definition of a functor, we see parts (1) and (2) tell us we can take a diagram and draw a similar one in the category $\mathcal{D}$ by sending all objects to their corresponding one under F_0 and similarly for morphisms under F_1 to get the diagram



Parts (3) and (4) tell us if the diagram in $\mathcal{C}$ is commutative, then the diagram in $\mathcal{D}$ is also commutative. So in this case, we can know that $F_1(g) \circ F_1(f) = F_1(h)$.

This is why we can think of functors as given "something shaped like $\mathcal{C}$" in $\mathcal{D}$, as the diagram shape is preserved. However, we do note the diagram can get "squished" – for example, if we have a functor from some category $\mathcal{C}$ into a category 1 that has only one object $\{*\}$ and only the identity morphism, we must have $F_0(a) = F_0(b) = F_0(c)$, and similar for the morphisms.

Now, we give a bunch of examples of functors. Our first class of functors are often called "forgetful functors". Recall how that our definition of a poset consists of an underlying set and a relation following some rules, and our morphisms of posets, i.e. monotone maps, were set theoretic functions following some additional rules. Therefore, we can make a functor U from the category of posets to the category of sets which "forgets" these additional rules as follows.

$$\begin{aligned} U &: \mathsf{Poset} \to \mathsf{Set} \\ U_0(X) &= |X|, \text{the underlying set of } X \\ U_1(f) &= f. \end{aligned}$$

We see that this assignment preserves the identity and composition as composition is the same in both categories – function composition as set-theoretic functions.

We can also do this for monoids, mapping monoids to their underlying set and monoid morphisms to their underlying set theoretic functions, and so we also have a forgetful functor $\mathsf{Monoid} \to \mathsf{Set}$. A slightly more interesting example is that of graphs, as we now have two choices for how we can map to set. Given a graph we can either map it to the set of its vertices, or the set of its edges. For a morphism of graphs, we then either take the function on the vertices, or the function on the edges as the set-theoretic function to map our graph morphism to.

We also have an example of a functor in our set-theoretic semantics for simple type theory. Recall that STTCTX is the category whose objects are contexts in STT, and morphisms are substitution functions as in homework 2 problem 2. We know that

$$[\![\delta \circ \gamma]\!] = [\![\delta]\!] \circ [\![\gamma]\!]$$

which is exactly saying that this denotation functor preserves composition. For the identity, we can easily show that

$$[\![\mathrm{Id}_\Gamma]\!] = \mathrm{Id}_{[\![\Gamma]\!]} : [\![\Gamma]\!] \to [\![\Gamma]\!]$$

is in fact the identity.

A more semantic example, FinList, is defined as follows.

$$\begin{aligned}
&\mathrm{FinList} : \mathsf{Set} \to \mathsf{Set} \\
&\mathrm{FinList}_0(X) = X^* := \{\text{set of finite sequences with elements in } X\} \\
&\mathrm{FinList}_1(f) = \text{map } f \text{ over the sequence,} \\
&\qquad\qquad \text{i.e. } \mathrm{FinList}_1(f)(x_1, x_2, \ldots, x_n) = (f(x_1), f(x_2), \ldots, f(x_n)).
\end{aligned}$$

Proving that this is a functor is straightforward. For arbitrary $X \in \mathsf{Set}_0$, we see for an arbitrary sequence $(x_1, x_2, \ldots, x_n) \in X^*$, we have

$$\begin{aligned}
\mathrm{FinList}(\mathrm{Id}_X)(x_1, x_2, \ldots, x_n) &= (\mathrm{Id}_X(x_1), \mathrm{Id}_X(x_2), \ldots, \mathrm{Id}_X(x_n)) \\
&= (x_1, x_2, \ldots, x_n),
\end{aligned}$$

and so $\mathrm{FinList}(\mathrm{Id}_X) = \mathrm{FinList}(\mathrm{Id}_{X^*}) = \mathrm{FinList}(\mathrm{Id}_{\mathrm{FinList}(X)})$. Now, for arbitrary $X, Y, Z \in \mathsf{Set}_0$ and functions $f : X \to Y$, $g : Y \to Z$, we have for arbitrary sequence $(x_1, \ldots, x_n) \in X^*$ that

$$\begin{aligned}
\mathrm{FinList}(g \circ f)(x_1, \ldots, x_n) &= ((g \circ f)(x_1), \ldots, (g \circ f)(x_n)) \\
&= (g(f(x_1)), \ldots, g(f(x_n)) \\
&= \mathrm{FinList}(g)(f(x_1), \ldots, f(x_n)) \\
&= \mathrm{FinList}(g)(\mathrm{FinList}(f)(x_1, \ldots, x_n)).
\end{aligned}$$

To give a practical viewpoint, if we imagine f and g to be functions in a programming language operation on a list, we note that the left hand side can be executed by iterating over the list once and applying $g \circ f$ to each element, while the right hand

side can be executed by iterating over the list twice, first applying f then applying g. However, if we know the above, we can see that the programming language can make the optimization of using the left hand side as its executional semantics, saving an iteration over the list. We can define similar notions of map for lots of traditional datatypes, such as trees. This is also an example of an endofunctor, which is a function whose source and target categories are the same.

Another example of an endofunctor is the powerset functor, defined as follows.

$$\begin{aligned}\mathcal{P} : \mathsf{Set} &\to \mathsf{Set}\\ \mathcal{P}(X) &= \text{subsets of } X\\ \mathcal{P}(f : X \to Y) &= \text{image under } f \text{ function,}\\ &\qquad \text{i.e. } \mathcal{P}(f)(S \subseteq X) = \{y \in Y \mid \exists s \in S \text{ s.t. } f(s) = y\}.\end{aligned}$$

Again, we have to prove that this is a functor. The verification is again straightforward. We see for arbitrary $X \in \mathsf{Set}_0$ and $S \subseteq X$, we have

$$\begin{aligned}\mathcal{P}(\mathrm{Id}_X)(S) &= \{x \in X \mid \exists s \in S \text{ s.t. } \mathrm{Id}_X(s) = x\}\\ &= \{x \in X \mid \exists s \in S \text{ s.t. } s = x\}\\ &= S.\end{aligned}$$

For arbitrary $X, Y, Z \in \mathsf{Set}_0$, $f : X \to Y$, $g : Y \to Z$, and $S \subseteq X$, we have

$$\mathcal{P}(g \circ f)(S) = \{z \in Z \mid \exists s \in S \text{ s.t. } g \circ f(s) = z\} \tag{1}$$

and

$$\begin{aligned}\mathcal{P}(g)(\mathcal{P}(f)(S)) &= \{z \in Z \mid \exists t \in \mathcal{P}(f)(S) \text{ s.t. } g(t) = z\}\\ &= \{z \in Z \mid \exists t \in \{y \in Y \mid \exists s \in S \text{ s.t. } f(s) = y\} \text{ s.t. } g(t) = z\} \qquad (2)\\ &= \{z \in Z . \exists s \in S . g(f(s)) = z\}\\ &= \mathcal{P}(g \circ f)(S)\end{aligned}$$

As we see these sets on the right hand side are equal, as if $z \in Z$ is in the set of equation 1, i.e. has $s \in S$ such that $g \circ f(s) = z$ we can take $t = f(s)$ and $s = s$ to see z is in the set of equation 2, and if $z \in Z$ is in the set of equation 2, we have $t = f(s)$ and $z = g(t)$, and so $z = (g \circ f)(s)$. It follows that these two sets are equal, and thus the functor $\mathcal{P}$ respects composition.

Now, we give a slightly more complicated example based on differentiable functions. To do this, we first must cook up some categories to make this work. Define the category Diff_* by

$$\begin{aligned}\mathrm{Diff}_{*0} &:= \mathbb{R}\\ \mathrm{Diff}_{*1}^{r,s} &:= \{f : \mathbb{R} \to \mathbb{R} \mid f \text{ is differentiable and } f(r) = s\}\end{aligned}$$

with composition defined as function composition, so we get associativity for free. This works because the identity function is differentiable and the composition of

differentiable function is differentiable. For our second category, we will take the monoid $(\mathbb{R}, *, 1)$, meaning the monoid of $\mathbb{R}$ under multiplication, which passes to categories as a single object category whose morphisms are the numbers in $\mathbb{R}$ – denote this object by a. Our functor Deriv is then defined by

$$\begin{aligned} \text{Deriv} : \text{Diff}_* &\to (\mathbb{R}, *, 1) \\ \text{Deriv}_0(r) &= a \\ \text{Deriv}_1(f : r \to s) &:= f'(r). \end{aligned}$$

To prove this is a functor, we see for arbitrary $r \in \mathbb{R}$ that the identity function is Id, and so

$$\text{Deriv}_1(\text{Id}) = \text{Id}'(r) = 1 = \text{Id}_a.$$

For composition, we see for arbitrary $r, s, t \in \text{Diff}_{*0}$ and $f : r \to s$, $g : s \to t$, that

$$\begin{aligned} \text{Deriv}_1(g \circ f) &= (g \circ f)'(r) \\ &= g'(f(r))f'(r) && \text{chain rule} \\ &= g'(s)f'(r) && f(r) = s \text{ by definition of } \text{Diff}_{*1} \\ &= \text{Deriv}_1(g)\text{Deriv}_1(f). \end{aligned}$$

While this example is a bit silly, there is an important takeaway, namely that the definition of category and functor are extremely general and so can many unexpected things can fit into the definitions. Also note that this generalizes to multi-variable derivatives, see Riehl's textbook for more details.

Finally, we give some general examples where we classify functors between certain types of categories. First, recall how we originally motivated our definition of functor from monotone maps between preorders. We now show that in a sense we did this generalization properly by showing that functors between thin categories correspond to monotone maps. We have already seen, in our motivation for the definition of a functor, that functors between thin categories are monotone functions on the underlying preorder. Now, we want to show any monotone function can be used to define a functor. Given two preorders/thin categories $\mathcal{C}$ and $\mathcal{D}$, we see if we have a monotone function $f : \mathcal{C} \to \mathcal{D}$, we can define the functor

$$\begin{aligned} F : \mathcal{C} &\to \mathcal{D} \\ F_0(a) &= f(a) \\ F_1(a \leq b) &= F_0(a) \leq F_0(b). \end{aligned}$$

We see that f being monotone guarantees that the above is well-defined, as $f(a) \leq f(b)$ implies $F_0(a) \leq F_0(b)$ is indeed a morphism. We conclude that functors between thin categories are exactly monotone functions.

We can do a similar thing for monoids. Given two monoids $\mathcal{C}$ and $\mathcal{D}$, we see our functor F has no option for where it maps the object of $\mathcal{C}$ to – as $\mathcal{D}$ has one object. Our functor also must map $F(\text{Id}) = \text{Id}$ and for any $f, g \in \mathcal{C}_1$, $F(f \circ g) = F(f) \circ F(g)$. Recalling that $\mathcal{C}_1$ is exactly the elements of the underlying monoid, we see that this criterion exactly corresponds to the notion of a monoid homomorphism.

2 Duality

A natural question following our observation that functors between thin categories correspond to monotone functions between preorders is if there is a categorical characterization of antitone functions between preorders. There is, and just as in an antitone function the ordering is reversed, in a contravariant functor we will reverse the order of the morphisms.

Definition 2 (Contravariant functor)**.** *Suppose $\mathcal{C} \to \mathcal{D}$. A contravariant functor $F : \mathcal{C} \to \mathcal{D}$ is an object containing the following information:*

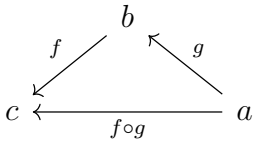
1. *An assignment of objects $F_0 : \mathcal{C}_0 \to \mathcal{D}_0$ which maps every object in $\mathcal{C}_0$ to an object in $\mathcal{D}_0$,*
2. *For all $a, b \in \mathcal{C}_0$, an assignment of morphisms $F_1^{a,b} : \mathcal{C}_1(a, b) \to \mathcal{D}_1(F_0(b), F_0(a))$,*
3. *such that F_1 preserves the identity morphism, i.e. for all objects $a \in \mathcal{C}_0$*

$$F_1^{a,a}(\mathrm{Id}_a) = \mathrm{Id}_{F_0(a)},$$

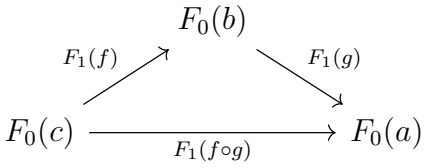
4. *and such that F_1 preserves composition, i.e. for all objects $a, b, c \in \mathcal{C}_0$ and morphisms $f \in \mathcal{C}_1(b, c)$ and $g \in \mathcal{C}_1(a, b)$, we have*

$$F_1^{a,c}(f \circ g) = F_1^{a,b}(g) \circ F_1^{b,c}(f)$$

Drawing out a diagram, if we have



this would get mapped to



under a contravariant functor, which allows us to see this reversal of direction of the morphisms.

To distinguish these from the other notion of functor, the other functors are called covariant functors. However, we typically do not consider contravariant functors as a separate notion from covariant ones. The reason for this is that using a construction called the opposite category, we can turn every contravariant functor into a covariant one by changing the source category.

Definition 3 (Opposite category)**.** *Suppose $\mathcal{C}$ is a category. We define the opposite category of $\mathcal{C}$, denoted by $\mathcal{C}^{\mathrm{op}}$ or $\mathcal{C}^{\mathrm{o}}$, to be the category*

$$\mathcal{C}_0^{\mathrm{op}} = \mathcal{C}_0$$
$$\mathcal{C}_1^{\mathrm{op}}(a,b) = \mathcal{C}_1(b,a)$$

and composition $f \circ^{\mathrm{op}} g$ in the opposite category by $g \circ f$.

The reason this is in fact a category is the same reason as why we could obtain a reverse order given any preorder. As a miscellaneous remark note that $\mathcal{C}^{\mathrm{opop}} = \mathcal{C}$, as reversing the direction twice is the same as doing nothing.

Using this opposite category we make the following observation.

Theorem 1. *A contravariant functor $\mathcal{C} \to \mathcal{D}$ is just a covariant functor $\mathcal{C}^{\mathrm{op}} \to \mathcal{D}$.*

Proof. Suppose F is a contravariant functor $\mathcal{C} \to \mathcal{D}$. We see criterion 2 of definition 2 tells us F gives assignments

$$F_1^{a,b} : \mathcal{C}_1(a,b) \to \mathcal{D}_1(F_0(b), F_0(a)).$$

But $\mathcal{C}_1(a,b) = \mathcal{C}_1^{\mathrm{op}}(b,a)$, so this functor immediately gives a covariant assignment of morphisms.

To check composition, we see for $a,b,c \in \mathcal{C}_0$ and morphisms $f \in \mathcal{C}_1(b,c)$ and $g \in \mathcal{C}_1(a,b)$ that

$$F_1^{a,c}(f \circ g) = F_1^{a,b}(g) \circ F_1^{b,c}(f)$$

Again, using that $\mathcal{C}_1(b,c) = \mathcal{C}_1^{\mathrm{op}}(c,b)$ and $\mathcal{C}_1(a,b) = \mathcal{C}_1^{\mathrm{op}}(b,a)$, we see the above is the same as saying

$$F_1^{c,a}(g \circ^{\mathrm{op}} f) = F_1^{a,b}(g) \circ F_1^{b,c}(f)$$

which tells us F_1 repsects composition as a covariant functor from $\mathcal{C}^{\mathrm{op}}$. □

This concept is called duality. Any definition made for a category can also be applied to the opposite category, which gives a dual definition. Additionally, as long as the definition is categorical in nature, any proofs made about the defintion carry over to the dual definition as well – the opposite category is itself a category after all.

To give another example of duality, recall that a monomorphism in a category $\mathcal{C}$ is a morphism $f : a \to b$ such that for all morphisms $x_1, x_2 : x \to a$, if $f \circ x_1 = f \circ x_2$ then $x_1 = x_2$. We can dualize this to get the following.

Definition 4 (Epimorphism)**.** *An epimorphism in $\mathcal{C}$ is a morphism which is a monomorphism in $\mathcal{C}^{\mathrm{op}}$.*

However, in terms of understanding this definition is not very helpful, so it often helps to think about unraveling this definition similarly to how we did for contravariant functors. $f : a \to b$ is a monomorphism in $\mathcal{C}^{\mathrm{op}}$ provided that for all morphisms

$x_1, x_2 \in \mathcal{C}_1^{\text{op}}(x, a)$, if $f \circ^{\text{op}} x_1 = f \circ^{\text{op}} x_2$, then $x_1 = x_2$. Undoing all the ops, the definition becomes that $f : a \to b$ is a epimorphism in $\mathcal{C}$ provided that for all morphisms $x_1, x_2 \in \mathcal{C}_1(a, x)$, if $x_1 \circ f = x_2 \circ f$, then $x_1 = x_2$, which is much easier to reason with in practice.

Note that in Set, epimorphisms are surjective, in a similar vein to how monomorphisms were injective.

Now, we give a couple examples of how proofs carry over to dual objects. Consider the following.

Theorem 2 (Monomorphisms compose)**.** *Suppose $\mathcal{C}$ is a category, and morphisms $f : a \to b$ and $g : b \to c$ in $\mathcal{C}$ are monomorphisms. Then $g \circ f$ is a monomorphism.*

Proof. Fix arbitrary $x_1, x_2 : x \to a$ such that $(g \circ f) \circ x_1 = (g \circ f) \circ x_2$. We wish to show that $x_1 = x_2$.

$$\begin{aligned} (g \circ f) \circ x_1 &= (g \circ f) \circ x_2 && \\ g \circ (f \circ x_1) &= g \circ (f \circ x_2) && \text{associativity} \\ f \circ x_1 &= f \circ x_2 && g \text{ is a monomorphism} \\ x_1 &= x_2 && f \text{ is a monomorphism.} \end{aligned}$$

□

As an immediate corollary, we get the following.

Corollary 1 (Epimorphisms compose)**.** *Suppose $\mathcal{C}$ is a category, and morphisms $f : a \to b$ and $g : b \to c$ in $\mathcal{C}$ are epimorphisms. Then $g \circ f$ is an epimorphism.*

Proof. By definition 4, we know f and g are monomorphisms in $\mathcal{C}^{\text{op}}$. By theorem 2 it follows that $f \circ^{\text{op}} g$ is an monomorphism in $\mathcal{C}^{\text{op}}$. But $f \circ^{\text{op}} g = g \circ f$, so we conclude $g \circ f$ is an epimorphism in $\mathcal{C}^{\text{op}}$. □

As an exercise, one can consider the following theorem which is similar to the order embedding problem on homework 2, and prove the analogous dual statement.

Theorem 3. *Suppose $\mathcal{C}$ is a category, and morphisms $f : a \to b$ and $g : b \to c$ such that $g \circ f$ is a monomorphism. Then f is a monomorphism.*

Corollary 2. *Suppose $\mathcal{C}$ is a category, and morphisms $f : a \to b$ and $g : b \to c$ such that $g \circ f$ is an epimorphism. Then g is an epimorphism.*

3 Constructions on categories

Having defined a notion of morphism for categories, we might hope to make a category of categories. We cannot make a small category of all categories, but we can for a fixed notion of small set, define a large category of all small categories:

$$\begin{aligned} \mathsf{Cat}_0 &:= \{\text{small categories}\} \\ \mathsf{Cat}_1(\mathcal{C}, \mathcal{D}) &:= \{\text{functors } F : \mathcal{C} \to \mathcal{D}\}. \end{aligned}$$

We of course need an identity morphism and composition, which are defined as follows

$$\begin{aligned}(\mathrm{Id}_{\mathcal{C}})_0(a) &= a\\ (\mathrm{Id}_{\mathcal{C}})_1(f) &= f\\ (F \circ G)_0(a) &= F_0(G_0(a))\\ (F \circ G)_1(f) &= F_1(G_1(f)).\end{aligned}$$

Verifying that these do in fact satisfy the category axioms is straightforward.

Working in this category, we can make a couple observations.

1. Let $\mathbf{1}$ denote the category consisting of 1 object $*$ with 1 (identity) morphism Id_*. Then for all categories $\mathcal{C}$, there is a unique functor $\mathcal{C} \to \mathbf{1}$, namely the functor which maps all objects to $*$ and all morphisms to Id_*. Going the other way, we see a functor $\mathbf{1} \to \mathcal{C}$ is just an object – the functor gets to choose the object $*$ gets mapped too, but as functors preserve identity morphism Id_* must get mapped to the corresponding identity morphism.

2. We can define a product of categories by

$$\begin{aligned}(\mathcal{C} \times \mathcal{D})_0 &= \mathcal{C}_0 \times \mathcal{D}_0\\ (\mathcal{C} \times \mathcal{D})_1((a,b),(a',b')) &= \mathcal{C}_1(a,a') \times \mathcal{D}_1(b,b')\end{aligned}$$

   with composition defined coordinate-wise. It is straightforward to check that this is indeed a category.

   This construction immediately gives two functors $\pi_1 : \mathcal{C} \times \mathcal{D} \to \mathcal{C}$ and $\pi_2 : \mathcal{C} \times \mathcal{D} \to \mathcal{D}$ that map out components, i.e.

$$\begin{aligned}\pi_1(a,b) &= a \qquad \pi_2(a,b) = b\\ \pi_1(f,g) &= f \quad \pi_2(f,g) = g.\end{aligned}$$

   This construction allows us to nicely define functors which takes two arguments, called bifunctors, as a functor from the cartesian product of the two categories.

3. Let $\mathbf{0}$ denote the empty category, consisting of no objects and thus no morphisms. We then get a unique functor $\mathbf{0} \to \mathcal{C}$ for any category $\mathcal{C}$, as there is nothing to define for our functor.

4. We can define a coproduct of categories by

$$\begin{aligned}(\mathcal{C} + \mathcal{D})_0 &= \mathcal{C}_0 \biguplus \mathcal{D}_0\\ (\mathcal{C} + \mathcal{D})_1(i_1 a, i_1 a') &= \mathcal{C}_1(a,a')\\ (\mathcal{C} + \mathcal{D})_1(i_2 b, i_2 b') &= \mathcal{D}_1(b,b')\\ (\mathcal{C} + \mathcal{D})_1(i_2 b, i_1 a') &= \emptyset\\ (\mathcal{C} + \mathcal{D})_1(i_1 a, i_2 b') &= \emptyset\end{aligned}$$

   and no morphisms for other cases.