Chapter 4

Basic category theory

"*... We know only a very few—and, therefore, very precious—schemes whose unifying powers cross many realms.*" – Marvin Minsky.[1]

Categories, or an equivalent notion, have already been secretly introduced as ologs. One can think of a category as a graph (as in Section 3.3) in which certain paths have been declared equivalent. (Ologs demand an extra requirement that everything in sight be readable in natural language, and this cannot be part of the mathematical definition of category.) The formal definition of category is given in Definition 4.1.1.1, but it will not be obviously the same as the "graph+path equivalences" notion; the latter was given in Definition 3.5.2.6 as the definition of a *schema*. Once we talk about how different categories can be compared using functors (Definition 4.1.2.1), and how different schemas can be compared using schema mappings (Definition 4.4.1.2), we will prove that the two notions are equivalent (Theorem 4.4.2.3).

4.1 Categories and Functors

In this section we give the standard definition of categories and functors. These, together with natural transformations (Section 4.3), form the backbone of category theory. We also give some examples.

4.1.1 Categories

In everyday speech we think of a category as a kind of thing. A category consists of a collection of things, all of which are related in some way. In mathematics, a category can also be construed as a collection of things and a type of relationship between pairs of such things. For this kind of thing-relationship duo to count as a category, we need to check two rules, which have the following flavor: every thing must be related to itself by simply being itself, and if one thing is related to another and the second is related to a third, then the first is related to the third. In a category, the "things" are called *objects* and the "relationships" are called *morphisms*.

In various places throughout this book so far we have discussed things of various sorts, e.g. sets, monoids, graphs. In each case we discussed how such things should be

[1] [Min, Problems of disunity, p. 126].

appropriately compared. In each case the "things" will stand as the objects and the "appropriate comparisons" will stand as the morphisms in the category. Here is the definition.

Definition 4.1.1.1. A *category* $\mathcal{C}$ is defined as follows: One announces some constituents (A. objects, B. morphisms, C. identities, D. compositions) and asserts that they conform to some laws (1. identity law, 2. associativity law). Specifically, one announces:

A. a collection $\text{Ob}(\mathcal{C})$, elements of which are called *objects*;

B. for every pair $x, y \in \text{Ob}(\mathcal{C})$, a set $\text{Hom}_{\mathcal{C}}(x, y) \in \textbf{Set}$. It is called the *hom-set from x to y*; its elements are called *morphisms from x to y*; [2]

C. for every object $x \in \text{Ob}(\mathcal{C})$, a specified morphism denoted $\text{id}_x \in \text{Hom}_{\mathcal{C}}(x, x)$ called *the identity morphism on x*; and

D. for every three objects $x, y, z \in \text{Ob}(\mathcal{C})$, a function

$$\circ\colon \text{Hom}_{\mathcal{C}}(y, z) \times \text{Hom}_{\mathcal{C}}(x, y) \to \text{Hom}_{\mathcal{C}}(x, z),$$

called *the composition formula*.

Given objects $x, y \in \text{Ob}(\mathcal{C})$, we can denote a morphism $f \in \text{Hom}_{\mathcal{C}}(x, y)$ by $f\colon x \to y$; we say that x is the *domain* of f and that y is the *codomain* of f. Given also $g\colon y \to z$, the composition formula is written using infix notation, so $g \circ f\colon x \to z$ means $\circ(g, f) \in \text{Hom}_{\mathcal{C}}(x, z)$.

One asserts that the following law holds:

1. for every $x, y \in \text{Ob}(\mathcal{C})$ and every morphism $f\colon x \to y$, we have

$$f \circ \text{id}_x = f \qquad \text{and} \qquad \text{id}_y \circ f = f;$$

and;

2. if $w, x, y, z \in \text{Ob}(\mathcal{C})$ are any objects and $f\colon w \to x$, $g\colon x \to y$, and $h\colon y \to z$ are any morphisms, then the two ways to compose are the same:

$$(h \circ g) \circ f = h \circ (g \circ f) \in \text{Hom}_{\mathcal{C}}(w, z).$$

Remark 4.1.1.2. There is perhaps much that is unfamiliar about Definition 4.1.1.1 but there is also one thing that is strange about it. The objects $\text{Ob}(\mathcal{C})$ of $\mathcal{C}$ are said to be a "collection" rather than a set. This is because we sometimes want to talk about the category of all sets, in which every possible set is an objects, and if we try to say that the collection of sets is itself, we run into Russell's paradox. Modeling this was a sticking point in the foundations of category theory, but it was eventually fixed by Grothendieck's notion of expanding universes. Roughly the idea is to choose some huge set κ (with certain properties making it a *universe*), to work entirely inside of it when possible, and to call anything in that world *κ-small* (or just *small* if κ is clear from context). When we need to look at κ itself, we choose an even bigger universe κ' and work entirely within it.

A category in which the collection $\text{Ob}(\mathcal{C})$ is a set (or in the above language, a small set) is called a *small category*. From here on out we will not take care of the difference, referring to $\text{Ob}(\mathcal{C})$ as a set. We do not think this will do any harm to scientists using category theory, at least not in the beginning phases of their learning.

[2] The reason for the notation Hom and the word *hom-set* is that morphisms are often called *homomorphisms*, e.g. in group theory.

Example 4.1.1.3 (The category **Set** of sets). Chapter 2 was all about the category of sets, denoted **Set**. The objects are the sets and the morphisms are the functions; we even used the current notation, referring to the set of functions $X \to Y$ as $\mathrm{Hom}_{\mathbf{Set}}(X, Y)$. The composition formula $\circ$ is given by function composition, and for every set X, the identity function $\mathrm{id}_X \colon X \to X$ serves as the identity morphism for $X \in \mathrm{Ob}(\mathbf{Set})$. The two laws clearly hold, so **Set** is indeed a category.

Example 4.1.1.4 (The category **Fin** of finite sets). Inside the category **Set** is a *subcategory* $\mathbf{Fin} \subseteq \mathbf{Set}$, called the *category of finite sets*. Whereas an object $S \in \mathrm{Ob}(\mathbf{Set})$ is a set that can have arbitrary cardinality, we define **Fin** such that its objects include all (and only) the sets S with finitely many elements, i.e. $|S| = n$ for some natural number $n \in \mathbb{N}$. Every object of **Fin** is an object of **Set**, but not vice versa.

Although **Fin** and **Set** have a different collection of objects, their morphisms are in some sense "the same". For any two finite sets $S, S' \in \mathrm{Ob}(\mathbf{Fin})$, we can also think of $S, S' \in \mathrm{Ob}(\mathbf{Set})$, and we have

$$\mathrm{Hom}_{\mathbf{Fin}}(S, S') = \mathrm{Hom}_{\mathbf{Set}}(S, S').$$

That is a morphism in **Fin** between finite sets S and S' is simply a function $f \colon S \to S'$.

Example 4.1.1.5 (The category **Mon** of monoids). We defined monoids in Definition 3.1.1.1 and monoid homomorphisms in Definition 3.1.4.1. Every monoid $\mathcal{M} := (M, e, \star_M)$ has an identity homomorphism $\mathrm{id}_{\mathcal{M}} \colon \mathcal{M} \to \mathcal{M}$, given by the identity function $\mathrm{id}_M \colon M \to M$. To compose two monoid homomorphisms $f \colon \mathcal{M} \to \mathcal{M}'$ and $g \colon \mathcal{M}' \to \mathcal{M}''$, we compose their underlying functions $f \colon M \to M'$ and $g \colon M' \to M''$, and check that the result $g \circ f$ is a monoid homomorphism. Indeed,

$$g \circ f(e) = g(e') = e''$$

$$g \circ f(m_1 \star_M m_2) = g(f(m_1) \star_{M'} f(m_2)) = g \circ f(m_1) \star_{M''} g \circ f(m_2).$$

It is clear that the two laws hold, so **Mon** is a category.

Exercise 4.1.1.6 (The category **Grp** of groups). Suppose we set out to define a category **Grp**, having groups as objects and group homomorphisms as morphisms, see Definition 3.2.1.16. Show (to the level of detail of Example 4.1.1.5) that the rest of the conditions for **Grp** to be a category are satisfied. ◊

Exercise 4.1.1.7 (The category **PrO** of preorders). Suppose we set out to define a category **PrO**, having preorders as objects and preorder homomorphisms as morphisms (see Definition 3.4.4.1). Show (to the level of detail of Example 4.1.1.5 that the rest of the conditions for **PrO** to be a category are satisfied. ◊

Example 4.1.1.8 (Non-category 1). So what's not a category? Two things can go wrong: either one fails to specify all the relevant constituents (A, B, C, D from Definition 4.1.1.1, or the constituents do not obey the laws (1, 2).

Let G be the following graph,

$$G = \boxed{\begin{tikzcd} \overset{a}{\bullet} \arrow[r, "f"] & \overset{b}{\bullet} \arrow[r, "g"] & \overset{c}{\bullet} \end{tikzcd}}.$$

Suppose we try to define a category $\mathcal{G}$ by faithfully recording vertices as objects and arrows as morphisms. Will that be a category?

Following that scheme, we put $\mathrm{Ob}(\mathcal{G}) = \{a, b, c\}$. For all 9 pairs of objects we need a hom-set. Say

$$\begin{array}{lll}
\mathrm{Hom}_{\mathcal{G}}(a,a) = \varnothing & \mathrm{Hom}_{\mathcal{G}}(a,b) = \{f\} & \mathrm{Hom}_{\mathcal{G}}(a,c) = \varnothing \\
\mathrm{Hom}_{\mathcal{G}}(b,a) = \varnothing & \mathrm{Hom}_{\mathcal{G}}(b,b) = \varnothing & \mathrm{Hom}_{\mathcal{G}}(b,c) = \{g\} \\
\mathrm{Hom}_{\mathcal{G}}(c,a) = \varnothing & \mathrm{Hom}_{\mathcal{G}}(c,b) = \varnothing & \mathrm{Hom}_{\mathcal{G}}(c,c) = \varnothing
\end{array}$$

If we say we are done, the listener should object that we have given neither identities nor a composition formula. In fact, it is impossible to give identities under our scheme, because e.g. $\mathrm{Hom}_{\mathcal{G}}(a,a) = \varnothing$.

Suppose we fix that problem, adding an element to each of our "diagonals" so that

$$\mathrm{Hom}_{\mathcal{G}}(a,a) = \{\mathrm{id}_a\}, \qquad \mathrm{Hom}_{\mathcal{G}}(b,b) = \{\mathrm{id}_b\}, \qquad \text{and} \qquad \mathrm{Hom}_{\mathcal{G}}(c,c) = \{\mathrm{id}_c\}.$$

What about a composition formula? We need a function $\mathrm{Hom}_{\mathcal{G}}(a,b) \times \mathrm{Hom}_{\mathcal{G}}(b,c) \to \mathrm{Hom}_{\mathcal{G}}(a,c)$, but the domain is nonempty and the codomain is empty; there is no such function.

Again, we must make a change, adding an element to make

$$\mathrm{Hom}_{\mathcal{G}}(a,c) = \{h\}.$$

We would now say $g \circ f = h$. Finally, this does the trick and we have a category. A computer could check this quickly, as can someone with good intuition for categories; for everyone else, it may be a painstaking process involving determining whether there is a unique composition formula for each of the 27 pairs of hom-sets and whether the associative law holds in the 81 necessary cases. Luckily this computation is "sparse" (lots of $\varnothing$'s), so it's not as bad as it first seems.

Redrawing all the morphisms as arrows, our graph has become:

$$G = \boxed{\begin{tikzcd}
\overset{a}{\bullet} \arrow[loop left, "\mathrm{id}_a"] \arrow[r, "f"] \arrow[rr, bend right, "h"'] & \overset{b}{\bullet} \arrow[loop above, "\mathrm{id}_b"] \arrow[r, "g"] & \overset{c}{\bullet} \arrow[loop right, "\mathrm{id}_c"]
\end{tikzcd}}$$

Example 4.1.1.9 (Non-category 2). In this example, we will make a faux-category $\mathcal{F}$ with one object and many morphisms. The problem here will be our composition formula.

Define $\mathcal{F}$ to have one object $\mathrm{Ob}(\mathcal{F}) = \{☺\}$, and $\mathrm{Hom}_{\mathcal{F}}(☺, ☺) = \mathbb{N}$. Define $\mathrm{id}_{☺} = 1 \in \mathbb{N}$. Define the composition formula $\circ\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ by $m \circ n = m^n$. This is a perfectly cromulent function, but it does not work right as a composition formula. Indeed, for the identity law to hold, we would need $m^1 = m = 1^m$, and one side of this is false. For the associativity law to hold, we would need $(m^n)^p = m^{(n^p)}$, but this is also not the case.

To fix this problem we have to completely revamp our composition formula. It would work to use multiplication, $m \circ n = m * n$. Then the identity law would read $1 * m = m = m * 1$, and that holds; and the associativity law would read $(m * n) * p = m * (n * p)$, and that holds.

Example 4.1.1.10 (The category of preorders with joins). Suppose that we are only interested in preorders $(X, \leqslant)$ for which every pair of elements has a join. We saw in Exercise 3.4.2.3 that not all preorders have this property. However we can create a category $\mathcal{C}$ in which every object does have this property. To begin we put $\mathrm{Ob}(\mathcal{C}) = \{(X, \leqslant) \in \mathrm{Ob}(\mathbf{PrO}) \mid (X, \leqslant) \text{ has all joins}\}$. But what about morphisms?

One option would be to put in no morphisms (other than identities), and to just consider this collection of objects as having no structure other than a set.

Another option would be to put in exactly the same morphisms as in **PrO**: for any objects $a, b \in \mathrm{Ob}(\mathcal{C})$ we consider a and b as regular old preorders, and put $\mathrm{Hom}_{\mathcal{C}}(a, b) := \mathrm{Hom}_{\mathbf{PrO}}(a, b)$. The resulting category of preorders with joins is called the *full subcategory of* **PrO** *spanned by the preorders with joins.*[3]

A third option, and the one perhaps that would jump out to a category theorist, is to take the choice about how we define our objects as a clue to how we should define our morphisms. Namely, if we are so interested in joins, perhaps we want joins to be preserved under morphisms. That is, if $f\colon (X, \leqslant_X) \to (Y, \leqslant_Y)$ is a morphism of preorders then for any join $w = x \vee x'$ in X we might want to enforce that $f(w) = f(x) \vee f(x')$ in Y. Thus a third possibility for the morphisms of $\mathcal{C}$ would be

$$\mathrm{Hom}_{\mathcal{C}}(a, b) := \{f \in \mathrm{Hom}_{\mathbf{PrO}}(a, b) \mid f \text{ preserves joins}\}.$$

One can check easily that the identity morphisms preserve joins and that compositions of join-preserving morphisms are join-preserving, so this version of homomorphisms makes for a well-defined category.

Example 4.1.1.11 (Category **FLin** of finite linear orders). We have a category **PrO** of preorders, and some of its objects are finite (nonempty) linear orders. Let **FLin** be the full subcategory of **PrO** spanned by the linear orders. That is, following Definition 3.4.4.1, given linear orders X, Y, every morphism of preorders $X \to Y$ counts as a morphism in **FLin**:

$$\mathrm{Hom}_{\mathbf{FLin}}(X, Y) = \mathrm{Hom}_{\mathbf{PrO}}(X, Y).$$

Exercise 4.1.1.12. Let **FLin** be the category of finite linear orders, defined in Example 4.1.1.11. For $n \in \mathbb{N}$, let $[n]$ be the linear order defined in Example 3.4.1.7. What are the cardinalities of the following sets:

a.) $\mathrm{Hom}_{\mathbf{FLin}}([0], [3])$;

b.) $\mathrm{Hom}_{\mathbf{FLin}}([3], [0])$;

c.) $\mathrm{Hom}_{\mathbf{FLin}}([2], [3])$;

d.) $\mathrm{Hom}_{\mathbf{FLin}}([1], [n])$?

e.) (Challenge) $\mathrm{Hom}_{\mathbf{FLin}}([m], [n])$?

It turns out that the category **FLin** of linear orders is sufficiently rich that much of algebraic topology (the study of arbitrary spaces, such as Mobius strips and 7-dimensional spheres) can be understood in its terms. See Example 4.6.1.6. ◊

Example 4.1.1.13 (Category of graphs). We defined graphs in Definition 3.3.1.1 and graph homomorphisms in Definition 3.3.3.1. To see that these are sufficient to form a category is considered routine to a seasoned category-theorist, so let's see why.

Since a morphism from $\mathcal{G} = (V, A, src, tgt)$ to $\mathcal{G}' = (V', A', src', tgt')$ involves two functions $f_0\colon V \to V'$ and $f_1\colon A \to A'$, the identity and composition formulas will simply arise from the identity and composition formulas for sets. Associativity will follow similarly. The only thing that needs to be checked, really, is that the composition of two such things, each satisfying (3.6), will itself satisfy (3.6). Just for completeness, we check that now.

[3]The definition of full subcategories will be given as Definition 4.6.3.1.

Suppose that $f = (f_0, f_1)\colon \mathcal{G} \to \mathcal{G}'$ and $g = (g_0, g_1)\colon \mathcal{G}' \to \mathcal{G}''$ are graph homomorphisms, where $\mathcal{G}'' = (V'', A'', src'', tgt'')$. Then in each diagram below

$$\begin{tikzcd} A \arrow[r, "f_1"] \arrow[d, "src"] & A' \arrow[r, "g_1"] \arrow[d, "src'"] & A'' \arrow[d, "src''"] \\ V \arrow[r, "f_0"'] & V' \arrow[r, "g_0"'] & V'' \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "f_1"] \arrow[d, "tgt"] & A' \arrow[r, "g_1"] \arrow[d, "tgt'"] & A'' \arrow[d, "tgt''"] \\ V \arrow[r, "f_0"'] & V' \arrow[r, "g_0"'] & V'' \end{tikzcd} \tag{4.1}$$

the left-hand square commutes because f is a graph homomorphism and the right-hand square commutes because g is a graph homomorphism. Thus the whole rectangle commutes, meaning that $g \circ f$ is a graph homomorphism, as desired.

We denote the category of graphs and graph homomorphisms by **Grph**.

Remark 4.1.1.14. When one is struggling to understand basic definitions, notation, and style, a phase which naturally occurs when learning new mathematics (or any new language), the above example will probably appear long and tiring. I'd say you've mastered the basics when the above example really does feel straightforward. Around this time, I imagine you'll begin to get a sense of the remarkable organisational potential of the categorical way of thinking.

Exercise 4.1.1.15. Let F be a vector field on $\mathbb{R}^2$. Recall that for two points $x, x' \in \mathbb{R}^2$, any curve C with endpoints x and x', and any parameterization $r\colon [a, b] \to C$, the line integral $\int_C F(r) \cdot dr$ returns a real number. It does not depend on r, except its orientation (direction). Therefore, if we think of C has having an orientation, say going from x to x', then $\int_C F$ is a well-defined real number. If C goes from x to x', let's suggestively write $C\colon x \to x'$. Define an equivalence relation $\sim$ on the set of oriented curves in $\mathbb{R}^2$ by saying $C \sim C'$ if

- C and C' start at the same point,
- C and C' end at the same point, and
- $\int_C F = \int_{C'} F$.

Suppose we try to make a category $\mathcal{C}_F$ as follows. Put $\mathrm{Ob}(\mathcal{C}_F) = \mathbb{R}^2$, and for every pair of points $x, x' \in \mathbb{R}^2$, let $\mathrm{Hom}_{\mathcal{C}_F}(x, x') = \{C\colon x \to x'\}/\sim$, where $C\colon x \to x'$ is an oriented curve and $\sim$ means "same line integral", as explained above.

Is there an identity morphism and a composition formula that will make $\mathcal{C}_F$ into a category? ◊

4.1.1.16 Isomorphisms

In any category we have a notion of isomorphism between objects.

Definition 4.1.1.17. Let $\mathcal{C}$ be a category and let $X, Y \in \mathrm{Ob}(\mathcal{C})$ be objects. An *isomorphism f from X to Y* is a morphism $f\colon X \to Y$ in $\mathcal{C}$, such that there exists a morphism $g\colon Y \to X$ in $\mathcal{C}$ such that

$$g \circ f = \mathrm{id}_X \qquad \text{and} \qquad f \circ g = \mathrm{id}_Y.$$

In this case we say that the morphism f is *invertible* and that g is the *inverse* of f. We may also say that the objects X and Y are *isomorphic*.

Example 4.1.1.18. If $\mathcal{C} = \textbf{Set}$ is the category of sets, then the above definition coincides precisely with the one given in Definition 2.1.2.8.

Exercise 4.1.1.19. Suppose that $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ are graphs and that $f = (f_0, f_1) \colon G \to G'$ is a graph homomorphism (as in Definition 3.3.3.1).

a.) If f is an isomorphism in **Grph**, does this imply that $f_0 \colon V \to V'$ and $f_1 \colon A \to A'$ are isomorphisms in **Set**?

b.) If so, why; and if not, show a counterexample (where f is an isomorphism but either f_0 or f_1 is not).

◊

Exercise 4.1.1.20. Suppose that $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ are graphs and that $f = (f_0, f_1) \colon G \to G'$ is a graph homomorphism (as in Definition 3.3.3.1).

a.) If $f_0 \colon V \to V'$ and $f_1 \colon A \to A'$ are isomorphisms in **Set**, does this imply that f is an isomorphism in **Grph**?

b.) If so, why; and if not, show a counterexample (where f_0 and f_1 are isomorphisms but f is not).

◊

Lemma 4.1.1.21. *Let $\mathcal{C}$ be a category and let $\sim$ be the relation on* $\mathrm{Ob}(\mathcal{C})$ *given by saying $X \sim Y$ iff X and Y are isomorphic. Then $\sim$ is an equivalence relation.*

Proof. The proof of Lemma 2.1.2.12 can be mimicked in this more general setting.

□

4.1.1.22 Another viewpoint on categories

Here is an alternate definition of category, using the work we did in Chapter 2.

Exercise 4.1.1.23. Suppose we begin our definition of category as follows.

A *category*, $\mathcal{C}$ consists of a sequence $(\mathrm{Ob}(\mathcal{C}), \mathrm{Hom}_{\mathcal{C}}, dom, cod, \mathrm{ids}, \circ)$, where

1. $\mathrm{Ob}(\mathcal{C})$ is a set,[4]

2. $\mathrm{Hom}_{\mathcal{C}}$ is a set, and $dom, cod \colon \mathrm{Hom}_{\mathcal{C}} \to \mathrm{Ob}(\mathcal{C})$ are functions,

3. $\mathrm{ids} \colon \mathrm{Ob}(\mathcal{C}) \to \mathrm{Hom}_{\mathcal{C}}$ is a function, and

[4]See Remark 4.1.1.2.

4. $\circ$ is a function as depicted in the commutative diagram below

$$\begin{tikzcd}
\mathrm{Hom}_{\mathcal{C}} \arrow[rrr, bend left, "cod"] \arrow[ddd, bend right, "dom"'] & & \checkmark & \\
\checkmark & \mathrm{Hom}_{\mathcal{C}} \times_{\mathrm{Ob}(\mathcal{C})} \mathrm{Hom}_{\mathcal{C}} \arrow[ul, "\circ"'] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{Hom}_{\mathcal{C}} \arrow[r, "cod"'] \arrow[d, "dom"] & \mathrm{Ob}(\mathcal{C}) \\
& \mathrm{Hom}_{\mathcal{C}} \arrow[r, "cod"'] \arrow[d, "dom"] & \mathrm{Ob}(\mathcal{C}) & \\
& \mathrm{Ob}(\mathcal{C}) & &
\end{tikzcd} \tag{4.2}$$

a.) Express the fact that for any $x \in \mathrm{Ob}(\mathcal{C})$ the morphism id_x points from x to x in terms of the functions id, dom, cod.

b.) Express the condition that composing a morphism f with an appropriate identity morphism yields f.

c.) Express the associativity law in these terms (Hint: Proposition 2.5.1.17 may be useful).

◊

Example 4.1.1.24 (Partial olog for a category). Below is an olog that captures some of the essential structures of a category.

$$\begin{tikzcd}
\boxed{\text{a morphism in } \mathcal{C}} \arrow[rrr, bend left, "\text{has as codomain}"] \arrow[ddd, bend right, "\text{has as domain}"'] & & \checkmark & \\
\checkmark & \boxed{\text{a pair } (g,f) \text{ of composable morphisms}} \arrow[ul, "\text{has as composition}"'] \arrow[r, "\text{yields as } g"] \arrow[d, "\text{yields as } f"'] \arrow[dr, phantom, "\lrcorner", very near start] & \boxed{\text{a morphism in } \mathcal{C}} \arrow[r, "\text{has as codomain}"'] \arrow[d, "\text{has as domain}"] & \boxed{\text{an object of } \mathcal{C}} \\
& \boxed{\text{a morphism in } \mathcal{C}} \arrow[r, "\text{has as codomain}"'] \arrow[d, "\text{has as domain}"] & \boxed{\text{an object of } \mathcal{C}} & \\
& \boxed{\text{an object of } \mathcal{C}} & &
\end{tikzcd} \tag{4.3}$$

Missing from (4.3) is the notion of identity morphism (as an arrow from ⌜an object of $\mathcal{C}$⌝ to ⌜a morphism in $\mathcal{C}$⌝) and the associated path equivalences, as well as the identity and associativity laws. All of these can be added to the olog, at the expense of some clutter.

Remark 4.1.1.25. Perhaps it is already clear that category theory is very interconnected. It may feel like everything relates to everything, and this feeling may intensify as you go on. However, the relationships between different notions are rigorously defined, and not random. Moreover, almost everything presented in this book can be formalized in a proof system like Coq (the most obvious exceptions being things like the readability requirement of ologs and the modeling of scientific applications).

Whenever you feel cognitive vertigo, look to formal definitions as the ground of your understanding. It is good practice to make sure that the intuition you've developed actually "touches down" on that ground, i.e. that your way of thinking can be built up solidly from the foundational definitions.

4.1.2 Functors

A category $\mathcal{C} = (\mathrm{Ob}(\mathcal{C}), \mathrm{Hom}_{\mathcal{C}}, dom, cod, \mathrm{ids}, \circ)$, involves a set of objects, a set of morphisms, a notion of domains and codomains, a notion of identity morphisms, and a composition formula. For two categories to be comparable, these various components should be appropriately comparable.

Definition 4.1.2.1. Let $\mathcal{C}$ and $\mathcal{C}'$ be categories. A *functor F from $\mathcal{C}$ to $\mathcal{C}'$*, denoted $F\colon \mathcal{C} \to \mathcal{C}'$, is defined as follows: One announces some constituents (A. on-objects part, B. on-morphisms part) and asserts that they conform to some laws (1. preservation of identities, 2. preservation of composition). Specifically, one announces

A. a function $\mathrm{Ob}(F)\colon \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C}')$, which we sometimes denote simply by $F\colon \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C}')$; and

B. for every pair of objects $c, d \in \mathrm{Ob}(\mathcal{C})$, a function

$$\mathrm{Hom}_F(c,d)\colon \mathrm{Hom}_{\mathcal{C}}(c,d) \to \mathrm{Hom}_{\mathcal{C}'}(F(c), F(d)),$$

which we sometimes denote simply by $F\colon \mathrm{Hom}_{\mathcal{C}}(c,d) \to \mathrm{Hom}_{\mathcal{C}'}(F(c), F(d))$.

One asserts that the following laws hold:

1. Identities are preserved by F. That is, for any object $c \in \mathrm{Ob}(\mathcal{C})$, we have $F(\mathrm{id}_c) = \mathrm{id}_{F(c)}$; and

2. Composition is preserved by F. That is, for any objects $b, c, d \in \mathrm{Ob}(\mathcal{C})$ and morphisms $g\colon b \to c$ and $h\colon c \to d$, we have $F(h \circ g) = F(h) \circ F(g)$.

Example 4.1.2.2 (Monoids have underlying sets). Recall from Definition 3.1.1.1 that if $\mathcal{M} = (M, e, \star)$ is a monoid, then M is a set. And recall from Definition 3.1.4.1 that if $f\colon \mathcal{M} \to \mathcal{M}'$ is a monoid homomorphism then $f\colon M \to M'$ is a function. Thus we have a functor

$$U\colon \mathbf{Mon} \to \mathbf{Set}$$

that takes every monoid to its underlying set and every monoid homomorphism to its underlying function.

Given two monoids $\mathcal{M} = (M, e, \star)$ and $\mathcal{M}' = (M', e', \star')$, there may be many functions from M to M' that do not arise from monoid homomorphisms. It is often useful to speak of such functions. For example, one could assign to every command in one video game V a command in another video game V', but this may not work in the "monoidy way" when performing a sequence of commands. By being able to speak of M as a set, or as $\mathcal{M}$ as a monoid, and understanding the relationship U between them, we can be clear about where we stand at all times in our discussion.

Example 4.1.2.3 (Groups have underlying monoids). Recall that a group is just a monoid $(M, e, \star)$ with the extra property that every element $m \in M$ has an inverse $m' \star m = e = m \star m'$. Thus to every group we can assign its *underlying monoid*. Similarly, a group homomorphism is just a monoid homomorphism of its underlying monoids. This means that there is a functor

$$U\colon \mathbf{Grp} \to \mathbf{Mon}$$

that sends every group or group homomorphism to its underlying monoid or monoid homomorphism. That identity and composition are preserved is obvious.

Slogan 4.1.2.4.

" *Out of all our available actions, some are reversable.* "

Application 4.1.2.5. Suppose you're a scientist working with symmetries. But then suppose that the symmetry breaks somewhere, or you add some extra observable which is not reversible under the symmetry. You want to seamlessly relax the requirement that every action be reversible without changing anything else. You want to know where you can go, or what's allowed. The answer is to simply pass from the category of groups (or group actions) to the category of monoids (or monoid actions).

We can also reverse this change of perspective. Recall that in Example 3.1.2.9 we discussed a monoid M controlling the actions of a video game character. The character position (P) could be moved up (u), moved down (d), or moved right (r). The path equivalences $P.u.d = P$ and $P.d.u = P$ imply that these two actions are mutually inverse, whereas moving right has no inverse. This, plus equivalences $P.r.u = P.u.r$ and $P.r.d = P.d.r$, defined a monoid M.

Inside M is a submonoid G, which includes just upward and downward movement. It has one object, just like M, i.e. $\mathrm{Ob}(M) = \{P\} = \mathrm{Ob}(G)$. But it has fewer morphisms. In fact there is a monoid isomorphism $G \cong \mathbb{Z}$ because we can assign to any movement in G the number of ups, e.g. $P.u.u.u.u.u$ is assigned the integer 5, $P.d.d.d$ is assigned the integer -3, and $P.d.u.u.d.d.u$ is assigned the integer $0 \in \mathbb{Z}$. But $\mathbb{Z}$ is a group, because every integer has an inverse.

Thus we can consider G as a group $G_1 \in \mathrm{Ob}(\mathbf{Grp})$ or as a monoid $G_2 \in \mathrm{Ob}(\mathbf{Mon})$. It is better to consider G as a group, because groups are more structured than monoids. It's as though putting G in $\mathbf{Grp}$ gives it more "potential energy" than putting it in $\mathbf{Mon}$ — we can always "drop it down" from $\mathbf{Grp}$ to $\mathbf{Mon}$, but not vice versa. The way to make this precise is that we can make use of the functor $U\colon \mathbf{Grp} \to \mathbf{Mon}$ from Example 4.1.2.3 and find that $U(G_1) = G_2$. But to find a functor $F\colon \mathbf{Mon} \to \mathbf{Grp}$ such that $F(G_2) = G_1$ would be much more ad hoc.

The upshot is that we can use functors to compare groups and monoids.

◊◊

Example 4.1.2.6. Recall that we have a category **Set** of sets and a category **Fin** of finite sets. We said that **Fin** was a subcategory of **Set**. In fact we can think of this "subcategory" relationship in terms of functors, just like we thought of the "subset" relationship in terms of functions in Example 2.1.2.3. That is, if we have a subset

$S \subseteq S'$, then every element $s \in S$ is an element of S', so we make a function $f\colon S \to S'$ such that $f(s) = s \in S'$.

To give a functor $i\colon \mathbf{Fin} \to \mathbf{Set}$, we have to announce how it will work on objects and how it will work on morphisms. We begin by announcing a function $i\colon \mathrm{Ob}(\mathbf{Fin}) \to \mathrm{Ob}(\mathbf{Set})$. But that's easy because $\mathrm{Ob}(\mathbf{Fin}) \subseteq \mathrm{Ob}(\mathbf{Set})$, so we proceed as above: $i(S) = S$ for any $S \in \mathrm{Ob}(\mathbf{Fin})$. We also have announce, for each pair of objects $S, S' \in \mathrm{Ob}(\mathbf{Fin})$, a function

$$i\colon \mathrm{Hom}_{\mathbf{Fin}}(S, S') \to \mathrm{Hom}_{\mathbf{Set}}(S, S').$$

But again, that's easy because we know by definition (see Example 4.1.1.4) that these two sets are equal, $\mathrm{Hom}_{\mathbf{Fin}}(S, S') = \mathrm{Hom}_{\mathbf{Set}}(S, S')$. Hence we can simply take i to be the identity function on morphisms. It is easy to see that identites and compositions are preserved by i. Therefore, we have defined a functor i.

Exercise 4.1.2.7 (Forgetful functors between types of orders). A partial order is just a preorder with a special property. A linear order is just a partial order with a special property.

a.) Is there an "obvious" functor $\mathbf{FLin} \to \mathbf{PrO}$?

b.) Is there an "obvious" functor $\mathbf{PrO} \to \mathbf{FLin}$?

◊

Proposition 4.1.2.8 (Preorders to graphs)**.** *Let* $\mathbf{PrO}$ *be the category of preorders and* $\mathbf{Grph}$ *be the category of graphs. There is a functor* $P\colon \mathbf{PrO} \to \mathbf{Grph}$ *such that for any preorder* $\mathcal{X} = (X, \leqslant)$*, the graph* $P(\mathcal{X})$ *has vertices* X*.*

Proof. Given a preorder $\mathcal{X} = (X, \leqslant_X)$, we can make a graph $F(\mathcal{X})$ with vertices X and an arrow $x \to x'$ whenever $x \leqslant_X x'$, as in Remark 3.4.1.10. More precisely, the preorder $\leqslant_X$ is a relation, i.e. a subset $R_{\mathcal{X}} \subseteq X \times X$, which we think of as a function $i\colon R_{\mathcal{X}} \to X \times X$. Composing with projections $\pi_1, \pi_2\colon X \times X \to X$ gives us

$$src_{\mathcal{X}} := \pi_1 \circ i\colon R_{\mathcal{X}} \to X \qquad \text{and} \qquad tgt_{\mathcal{X}} := \pi_2 \circ i\colon R_{\mathcal{X}} \to X.$$

Then we put $F(\mathcal{X}) := (X, R_{\mathcal{X}}, src_{\mathcal{X}}, tgt_{\mathcal{X}})$. This gives us a function $F\colon \mathrm{Ob}(\mathbf{PrO}) \to \mathrm{Ob}(\mathbf{Grph})$.

Suppose now that $f\colon \mathcal{X} \to \mathcal{Y}$ is a preorder morphism (where $\mathcal{Y} = (Y, \leqslant_Y)$). This is a function $f\colon X \to Y$ such that for any $(x, x') \in X \times X$, if $x \leqslant_X x'$ then $f(x) \leqslant f(x')$. But that's the same as saying that there exists a dotted arrow making the following diagram of sets commute

$$\begin{tikzcd} R_{\mathcal{X}} \arrow[r] \arrow[d, dotted] & X \times X \arrow[d, "f \times f"] \\ R_{\mathcal{Y}} \arrow[r] & Y \times Y \end{tikzcd}$$

(Note that there cannot be two different dotted arrows making that diagram commute because $R_{\mathcal{Y}} \to Y \times Y$ is a monomorphism.) Our commutative square is precisely what's needed for a graph homomorphism, as shown in Exercise 3.3.3.7. Thus, we have defined F on objects and on morphisms. It is clear that F preserves identity and composition. □

Exercise 4.1.2.9. In Proposition 4.1.2.8 we gave a functor $P\colon \mathbf{PrO} \to \mathbf{Grph}$.

a.) Is every graph $G \in \mathrm{Ob}(\mathbf{Grph})$ in the image of P (or more precisely, is the function

$$\mathrm{Ob}(P) \colon \mathrm{Ob}(\mathbf{PrO}) \to \mathrm{Ob}(\mathbf{Grph})$$

surjective)?

b.) If so, why; if not, name a graph not in the image.

c.) Suppose that $G, H \in \mathrm{Ob}(\mathbf{Grph})$ are two graphs that are in the image of P. Is every graph homomorphism $f \colon G \to H$ in the image of Hom_P? In other words, does every graph homomorphism between G and H come from a preorder homomorphism?

◊

Remark 4.1.2.10. There is a functor $W \colon \mathbf{PrO} \to \mathbf{Set}$ sending $(X, \leqslant)$ to X. There is a functor $T \colon \mathbf{Grph} \to \mathbf{Set}$ sending (V, A, src, tgt) to V. When we understand the category of categories (Section 4.1.2.27), it will be clear that Proposition 4.1.2.8 can be summarized as a commutative triangle in **Cat**,

$$\begin{tikzcd}
\mathbf{PrO} \arrow[rr, "P"] \arrow[dr, "W"'] & & \mathbf{Grph} \arrow[dl, "T"] \\
& \mathbf{Set} &
\end{tikzcd}$$

Exercise 4.1.2.11 (Graphs to preorders). Recall from (2.3) that every function $f \colon A \to B$ has an image, $\mathrm{im}_f(A) \subseteq B$. Use this idea and Example 3.4.1.16 to construct a functor $Im \colon \mathbf{Grph} \to \mathbf{PrO}$ such that for any graph $G = (V, A, src, tgt)$, the preorder has elements given by the vertices of G (i.e. we have $Im(G) = (V, \leqslant_G)$, for some ordering $\leqslant_G$). ◊

Exercise 4.1.2.12. What is the preorder $Im(G)$ when $G \in \mathrm{Ob}(\mathbf{Grph})$ is the following graph?

$$G := \begin{tikzcd}
\overset{v}{\bullet} \arrow[r, "f"] & \overset{w}{\bullet} \arrow[r, bend left, "g"] \arrow[r, bend right, "h"'] & \overset{x}{\bullet} \\
\overset{y}{\bullet} \arrow[loop left, "i"] \arrow[r, bend left, "j"] & \overset{z}{\bullet} \arrow[l, bend left, "k"] &
\end{tikzcd}$$

◊

Exercise 4.1.2.13. Consider the functor $Im \colon \mathbf{Grph} \to \mathbf{PrO}$ constructed in Exercise 4.1.2.11.

a.) Is every preorder $\mathcal{X} \in \mathrm{Ob}(\mathbf{PrO})$ in the image of Im (or more precisely in the image of $\mathrm{Ob}(Im) \colon \mathrm{Ob}(\mathbf{Grph}) \to \mathrm{Ob}(\mathbf{PrO})$)?

b.) If so, why; if not, name a preorder not in the image.

c.) Suppose that $\mathcal{X}, \mathcal{Y} \in \mathrm{Ob}(\mathbf{PrO})$ are two preorders that are in the image of Im. Is every preorder morphism $f \colon \mathcal{X} \to \mathcal{Y}$ in the image of Hom_{Im}? In other words, does every preorder homomorphism between $\mathcal{X}$ and $\mathcal{Y}$ come from a graph homomorphism?

◊

Exercise 4.1.2.14. We have functors $P\colon \mathbf{PrO} \to \mathbf{Grph}$ and $Im\colon \mathbf{Grph} \to \mathbf{PrO}$.

a.) What can you say about $Im \circ P\colon \mathbf{PrO} \to \mathbf{PrO}$?

b.) What can you say about $P \circ Im\colon \mathbf{Grph} \to \mathbf{Grph}$?

◊

Exercise 4.1.2.15. Consider the functors $P\colon \mathbf{PrO} \to \mathbf{Grph}$ and $Im\colon \mathbf{Grph} \to \mathbf{PrO}$. And consider the chain graph $[n]$ of length n from Example 3.3.1.8 and the linear order $[n]$ of length n from Example 3.4.1.7. To differentiate the two, let's rename them for this exercise as $[n]_{\mathbf{Grph}} \in \mathrm{Ob}(\mathbf{Grph})$ and $[n]_{\mathbf{PrO}} \in \mathrm{Ob}(\mathbf{PrO})$. We see a similarity between $[n]_{\mathbf{Grph}}$ and $[n]_{\mathbf{PrO}}$, and we might hope that our functors help us formalize this similarity. That is, we might hope that one of the following hold:

$$P([n]_{\mathbf{PrO}}) \cong^{?} [n]_{\mathbf{Grph}} \qquad \text{or} \qquad Im([n]_{\mathbf{Grph}}) \cong^{?} [n]_{\mathbf{PrO}}.$$

Do either, both, or neither of these hold? ◊

Remark 4.1.2.16. In the course announcement for 18-S996, I wrote the following:

> It is often useful to focus ones study by viewing an individual thing, or a group of things, as though it exists in isolation. However, the ability to rigorously change our point of view, seeing our object of study in a different context, often yields unexpected insights. Moreover this ability to change perspective is indispensable for effectively communicating with and learning from others. It is the relationships between things, rather than the things in and by themselves, that are responsible for generating the rich variety of phenomena we observe in the physical, informational, and mathematical worlds.

This holds at many different levels. For example, one can study a group (in the sense of Definition 3.2.1.1) in isolation, trying to understand its subgroups or its automorphisms, and this is mathematically interesting. But one can also view it as a quotient of something else, or as a subgroup of something else. One can view the group as a monoid and look at monoid homomorphisms to or from it. One can look at the group in the context of symmetries by seeing how it acts on sets. These changes of viewpoint are all clearly and formally expressible within category theory. We know how the different changes of viewpoint compose and how they fit together in a larger context.

Exercise 4.1.2.17.

a.) Is the above quote also true in your scientific discipline of expertise? How so?

b.) Can you imagine a way that category theory can help catalogue the kinds of relationships or changes of viewpoint that exist in your discipline?

c.) What kinds of structures that you use often really deserve to be better formalized?

Keep this kind of question in mind for your final project. ◊

Example 4.1.2.18 (Free monoids). Let G be a set. We saw in 3.1.1.15 that $\mathrm{List}(G)$ is a monoid, called the free monoid on G. Given a function $f\colon G \to G'$, there is an induced function $\mathrm{List}(f)\colon \mathrm{List}(G) \to \mathrm{List}(G')$, and this preserves the identity element $[\ ]$ and concatenation of lists, so $\mathrm{List}(f)$ is a monoid homomorphism. It is easy to check that $\mathrm{List}\colon \mathbf{Set} \to \mathbf{Mon}$ is a functor.

Application 4.1.2.19. In Application 2.1.2.10 we discussed an isomorphism $\text{Nuc}_{\text{DNA}} \cong \text{Nuc}_{\text{RNA}}$ given by RNA transcription. Applying the functor List we get a function

$$\text{List}(\text{Nuc}_{\text{DNA}}) \xrightarrow{\cong} \text{List}(\text{Nuc}_{\text{RNA}}),$$

which will send sequences of DNA nucleotides to sequences of RNA nucleotides and vice versa. This is performed by polymerases.

◊◊

Exercise 4.1.2.20. Let $G = \{1, 2, 3, 4, 5\}, G' = \{a, b, c\}$, and let $f\colon G \to G'$ be given by the sequence (a, c, b, a, c).[5] Then if $L = [1, 1, 3, 5, 4, 5, 3, 2, 4, 1]$, what is $\text{List}(f)(L)$? ◊

Exercise 4.1.2.21. We can rephrase our notion of functor in terms compatible with Exercise 4.1.1.23. We would begin by saying that a functor $F\colon \mathcal{C} \to \mathcal{C}'$ consists of two functions,

$$\text{Ob}(F)\colon \text{Ob}(\mathcal{C}) \to \text{Ob}(\mathcal{C}') \qquad \text{and} \qquad \text{Hom}_F\colon \text{Hom}_{\mathcal{C}} \to \text{Hom}_{\mathcal{C}'},$$

which we call the *on-objects part* and the *on-morphisms part*, respectively. They must follow some rules, expressed by the commutativity of the following squares in **Set**:

$$\begin{tikzcd}
\text{Hom}_{\mathcal{C}} \arrow[r, "dom"] \arrow[d, "\text{Hom}_F"'] & \text{Ob}(\mathcal{C}) \arrow[d, "\text{Ob}(F)"] \\
\text{Hom}_{\mathcal{C}'} \arrow[r, "dom"'] & \text{Ob}(\mathcal{C}')
\end{tikzcd}
\qquad
\begin{tikzcd}
\text{Hom}_{\mathcal{C}} \arrow[r, "cod"] \arrow[d, "\text{Hom}_F"'] & \text{Ob}(\mathcal{C}) \arrow[d, "\text{Ob}(F)"] \\
\text{Hom}_{\mathcal{C}'} \arrow[r, "cod"'] & \text{Ob}(\mathcal{C}')
\end{tikzcd}
\tag{4.4}$$

$$\begin{tikzcd}
\text{Ob}(\mathcal{C}) \arrow[r, "\text{id}"] \arrow[d, "\text{Ob}(F)"'] & \text{Hom}_{\mathcal{C}} \arrow[d, "\text{Hom}_F"] \\
\text{Ob}(\mathcal{C}') \arrow[r, "\text{id}"'] & \text{Hom}_{\mathcal{C}'}
\end{tikzcd}
\qquad
\begin{tikzcd}
\text{Hom}_{\mathcal{C}} \times_{\text{Ob}(\mathcal{C})} \text{Hom}_{\mathcal{C}} \arrow[r, "\circ"] \arrow[d] & \text{Hom}_{\mathcal{C}} \arrow[d, "\text{Hom}_F"] \\
\text{Hom}_{\mathcal{C}'} \times_{\text{Ob}(\mathcal{C}')} \text{Hom}_{\mathcal{C}'} \arrow[r, "\circ"'] & \text{Hom}_{\mathcal{C}'}
\end{tikzcd}
\tag{4.5}$$

Where does the (unlabeled) left-hand function in the bottom right diagram come from? Hint: use Exercise 2.5.1.19.

Consider Diagram (4.2) and imagine it as though contained in a pane of glass. Then imagine a parallel pane of glass involving $\mathcal{C}'$ in place of $\mathcal{C}$ everywhere.

a.) Draw arrows from the $\mathcal{C}$ pane to the $\mathcal{C}'$ pane, each labeled $\text{Ob}(F)$ or Hom_F as seems appropriate.

b.) If F is a functor (i.e. satisfies (4.4) and (4.5)), do all the squares in your drawing commute?

c.) Does the definition of functor involve anything not captured in this setup?

◊

Example 4.1.2.22 (Paths-graph). Let $G = (V, A, src, tgt)$ be a graph. Then for any pair of vertices $v, w \in G$, there is a set $\text{Path}_G(v, w)$ of paths from v to w; see Definition 3.3.2.1.

[5] See Exercise 2.1.2.15 in case there is any confusion with this.

In fact there is a set Path_G and functions $\overline{src}, \overline{tgt} \colon \mathrm{Path}_G \to V$. That information is enough to define a new graph,

$$\mathrm{Paths}(G) := (V, \mathrm{Path}_G, \overline{src}, \overline{tgt}).$$

Moreover, given a graph homomorphism $f \colon G \to G'$, every path in G is sent under f to a path in G'. So $\mathrm{Paths} \colon \mathbf{Grph} \to \mathbf{Grph}$ is a functor.

Exercise 4.1.2.23.

a.) Consider the graph G from Example 3.3.3.3. Draw the paths-graph $\mathrm{Paths}(G)$ for G.

b.) Repeating the above exercise for G' from the same example would be hard, because the path graph $\mathrm{Paths}(G')$ has infinitely many arrows. However, the graph homomorphism $f \colon G \to G'$ does induce a morphism of paths-graphs $\mathrm{Paths}(f) \colon \mathrm{Paths}(G) \to \mathrm{Paths}(G')$, and it is possible to say how that acts on the vertices and arrows of $\mathrm{Paths}(G)$. Please do so.

c.) Given a graph homomorphism $f \colon G \to G'$ and two paths $p \colon v \to w$ and $q \colon w \to x$ in G, is it true that $\mathrm{Paths}(f)$ preserves the concatenation? What does that even mean?

◊

Exercise 4.1.2.24. Suppose that $\mathcal{C}$ and $\mathcal{D}$ are categories, $c, c' \in \mathrm{Ob}(\mathcal{C})$ are objects, and $F \colon \mathcal{C} \to \mathcal{D}$ is a functor. Suppose that c and c' are isomorphic in $\mathcal{C}$. Show that this implies that $F(c)$ and $F(c')$ are isomorphic in $\mathcal{D}$. ◊

Example 4.1.2.25. For any graph G, we can assign its set of loops $Eq(G)$ as in Exercise 3.3.1.12. This assignment is functorial in that given a graph homomorphism $G \to G'$ there is an induced function $Eq(G) \to Eq(G')$. Similarly, we can functorially assign the set of connected components of the graph, $Coeq(G)$. In other words $Eq \colon \mathbf{Grph} \to \mathbf{Set}$ and $Coeq \colon \mathbf{Grph} \to \mathbf{Set}$ are functors. The assignment of vertex set and arrow set are two more functors $\mathbf{Grph} \to \mathbf{Set}$.

Suppose you want to decide whether two graphs G and G' are isomorphic. Supposing that the graphs have thousands of vertices and thousands of arrows, this could take a long time. However, the functors above, in combination with Exercise 4.1.2.24 give us some things to try.

The first thing to do is to count the number of loops of each, because these numbers are generally small. If the number of loops in G is different than the number of loops in G' then because functors preserve isomorphisms, G and G' cannot be isomorphic. Similarly one can count the number of connected components, again generally a small number; if the number of components in G is different than the number of components in G' then $G \not\cong G'$. Similarly, one can simply count the number of vertices or the number of arrows in G and G'. These are all isomorphism invariants.

All this is a bit like trying to decide if a number is prime by checking if it's even, if its digits add up to a multiple of 3, or it ends in a 5; these tests do not determine the answer, but they offer some level of discernment.

Remark 4.1.2.26. In the introduction I said that functors allow ideas in one domain to be rigorously imported to another. Example 4.1.2.25 is a first taste. Because functors preserve isomorphisms, we can tell graphs apart by looking at them in a simpler category, **Set**. There is relatively simple theorem in **Set** that says that for different natural numbers m, n the sets $\underline{m}$ and $\underline{n}$ are never isomorphic. This theorem is transported via our four functors to four different theorems about telling graphs apart.

4.1.2.27 The category of categories

Recall from Remark 4.1.1.2 that a small category $\mathcal{C}$ is one in which $\mathrm{Ob}(\mathcal{C})$ is a set. We have not really been paying attention to this issue, and everything we have said so far works whether $\mathcal{C}$ is small or not. In the following definition we really ought to be a little more careful, so we are.

Proposition 4.1.2.28. *There exists a category, called* the category of small categories *and denoted* **Cat**, *in which the objects are the small categories and the morphisms are the functors,*

$$\mathrm{Hom}_{\mathbf{Cat}}(\mathcal{C}, \mathcal{D}) = \{F\colon \mathcal{C} \to \mathcal{D} \mid F \text{ is a functor}\}.$$

That is, there are identity functors, functors can be composed, and the identity and associativity laws hold.

Proof. We follow Definition 4.1.1.1. We have specified $\mathrm{Ob}(\mathbf{Cat})$ and $\mathrm{Hom}_{\mathbf{Cat}}$ already. Given a small category $\mathcal{C}$, there is an identity functor $\mathrm{id}_{\mathcal{C}}\colon \mathcal{C} \to \mathcal{C}$ that is identity on the set of objects and the set of morphisms. And given a functor $F\colon \mathcal{C} \to \mathcal{D}$ and a functor $G\colon \mathcal{D} \to \mathcal{E}$, it is easy to check that $G \circ F\colon \mathcal{C} \to \mathcal{E}$, defined by composition of functions $\mathrm{Ob}(G) \circ \mathrm{Ob}(F)\colon \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{E})$ and $\mathrm{Hom}_G \circ \mathrm{Hom}_F\colon \mathrm{Hom}_{\mathcal{C}} \to \mathrm{Hom}_{\mathcal{E}}$ (see Exercise 4.1.2.21), is a functor. For the same reasons, it is easy to show that functors obey the identity law and the composition formula. Therefore this specification of **Cat** satisfies the definition of being a category.

□

Example 4.1.2.29 (Categories have underlying graphs). Let $\mathcal{C} = (\mathrm{Ob}(\mathcal{C}), \mathrm{Hom}_{\mathcal{C}}, dom, cod, \mathrm{ids}, \circ)$ be a category (see Exercise 4.1.1.23). Then $(\mathrm{Ob}(\mathcal{C}), \mathrm{Hom}_{\mathcal{C}}, dom, cod)$ is a graph, which we will call the *graph underlying* $\mathcal{C}$ and denote by $U(\mathcal{C}) \in \mathrm{Ob}(\mathbf{Grph})$. A functor $F\colon \mathcal{C} \to \mathcal{D}$ induces a graph morphism $U(F)\colon U(\mathcal{C}) \to U(\mathcal{D})$, as seen in (4.4). So we have a functor,

$$U\colon \mathbf{Cat} \to \mathbf{Grph}.$$

Example 4.1.2.30 (Free category on a graph). In Example 4.1.2.22, we discussed a functor $\mathrm{Paths}\colon \mathbf{Grph} \to \mathbf{Grph}$ that considered all the paths in a graph G as the arrows of a new graph $\mathrm{Paths}(G)$. In fact, $\mathrm{Paths}(G)$ could be construed as a category, which we will denote $F(G) \in \mathrm{Ob}(\mathbf{Cat})$ and call *the free category generated by* G.

Here, the objects of the category $F(G)$ are the vertices of G. For any two vertices v, v' the hom-set $\mathrm{Hom}_{F(G)}(v, v')$ is the set of paths in G from v to v'. The identity elements are given by the trivial paths, and the composition formula is given by concatenation of paths.

To see that F is a functor, we need to see that a graph homomorphism $f\colon G \to G'$ induces a functor $F(f)\colon F(G) \to F(G')$. But this was shown in Exercise 4.1.2.23. Thus we have a functor

$$F\colon \mathbf{Grph} \to \mathbf{Cat}$$

called *the free category* functor.

Exercise 4.1.2.31. Let G be the graph depicted

$$\begin{tikzcd} \overset{v_0}{\bullet} \arrow[r, "e"] & \overset{v_1}{\bullet}, \end{tikzcd}$$

and let $[1] \in \mathrm{Ob}(\mathbf{Cat})$ denote the free category on G (see Example 4.1.2.30). We call $[1]$ the *free arrow category.*

a.) What are its objects?

b.) For every pair of objects in $[1]$, write down the hom-set.

◊

Exercise 4.1.2.32. Let G be the graph whose vertices are all cities in the US and whose arrows are airplane flights connecting cities. What idea is captured by the free category on G? ◊

Exercise 4.1.2.33. Let $F\colon \mathbf{Grph} \to \mathbf{Cat}$ denote the free category functor from Example 4.1.2.30, and let $U\colon \mathbf{Cat} \to \mathbf{Grph}$ denote the underlying graph functor from Example 4.1.2.29. We have seen the composition $U \circ F\colon \mathbf{Grph} \to \mathbf{Grph}$ before; what was it called? ◊

Exercise 4.1.2.34. Recall the graph G from Example 3.3.1.2. Let $\mathcal{C} = F(G)$ be the free category on G.

a.) What is $\mathrm{Hom}_{\mathcal{C}}(v, x)$?

b.) What is $\mathrm{Hom}_{\mathcal{C}}(x, v)$?

◊

Example 4.1.2.35 (Discrete graphs, discrete categories). There is a functor $Disc\colon \mathbf{Set} \to \mathbf{Grph}$ that sends a set S to the graph

$$Disc(S) := (S, \varnothing, !, !),$$

where $!\colon \varnothing \to S$ is the unique function. We call $Disc(S)$ the *discrete graph on the set S*. It is clear that a function $S \to S'$ induces a morphism of discrete graphs. Now applying the free category functor $F\colon \mathbf{Grph} \to \mathbf{Cat}$, we get the so-called *discrete category on the set S*, which we also might call $Disc\colon \mathbf{Set} \to \mathbf{Cat}$.

Exercise 4.1.2.36. Recall from (2.6) the definition of the set $\underline{n}$ for any natural number $n \in \mathbb{N}$, and let $D_n := Disc(\underline{n}) \in \mathrm{Ob}(\mathbf{Cat})$.

a.) List all the morphisms in D_4.

b.) List all the functors $D_3 \to D_2$.

◊

Exercise 4.1.2.37 (Terminal category). Let $\mathcal{C}$ be a category. How many functors are there $\mathcal{C} \to D_1$, where $D_1 := Disc(\underline{1})$ is the discrete category on one element? ◊

We sometimes refer to $Disc(\underline{1})$ as the *terminal category* (for reasons that will be made clear in Section 4.5.3), and for simplicity denote it by $\underline{1}$.

Exercise 4.1.2.38. If someone said "Ob is a functor from **Cat** to **Set**," what might they mean? ◊

4.2 Categories and functors commonly arising in mathematics

4.2.1 Monoids, groups, preorders, and graphs

We saw in Section 4.1.1 that there is a category **Mon** of monoids, a category **Grp** of groups, a category **PrO** of preorders, and a category **Grph** of graphs. In this section we show that each monoid $\mathcal{M}$, each group $\mathcal{G}$, and each preorder $\mathcal{P}$ can be considered as its own category. If each object in **Mon** is a category, we might hope that each morphism in **Mon** is just a functor, and this is true. The same holds for **Grp** and **PrO**. We will deal with graphs in Section 4.2.1.20.

4.2.1.1 Monoids as categories

In Example 3.1.2.9 we said that to olog a monoid, we should use only one box. And again in Example 3.5.3.3 we said that a monoid action could be captured by only one table. These ideas emanated from the understanding that a monoid is perfectly modeled as a category with one object.

Each monoid as a category with one object Let $(M, e, \star)$ be a monoid. We consider it as a category $\mathcal{M}$ with one object, $\mathrm{Ob}(\mathcal{M}) = \{\blacktriangle\}$, and

$$\mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle) := M.$$

The identity morphism $\mathrm{id}_\blacktriangle$ serves as the monoid identity e, and the composition formula

$$\circ\colon \mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle) \times \mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle) \to \mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle)$$

is given by $\star\colon M \times M \to M$. The associativity and identity laws for the monoid match precisely with the associativity and identity laws for categories.

If monoids are categories with one object, is there any categorical way of phrasing the notion of monoid homomorphism? Suppose that $\mathcal{M} = (M, e, \star)$ and $\mathcal{M}' = (M', e', \star')$. We know that a monoid homomorphism is a function $f\colon M \to M'$ such that $f(e) = e'$ and such that for every pair $m_0, m_1 \in M$ we have $f(m_0 \star m_1) = f(m_0) \star' f(m_1)$. What is a functor $\mathcal{M} \to \mathcal{M}'$?

Each monoid homomorphism as a functor between one-object categories Say that $\mathrm{Ob}(\mathcal{M}) = \{\blacktriangle\}$ and $\mathrm{Ob}(\mathcal{M}') = \{\blacktriangle'\}$; and we know that $\mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle) = M$ and $\mathrm{Hom}_{\mathcal{M}'}(\blacktriangle', \blacktriangle') = M'$. A functor $F\colon \mathcal{M} \to \mathcal{M}'$ consists first of a function $\mathrm{Ob}(\mathcal{M}) \to \mathrm{Ob}(\mathcal{M}')$, but these sets have only one element each, so there is nothing to say on that front. It also consists of a function $\mathrm{Hom}_{\mathcal{M}} \to \mathrm{hom}_{\mathcal{M}'}$ but that is just a function $M \to M'$. The identity and composition formulas for functors match precisely with the identity and composition formula for monoid homomorphisms, as discussed above. Thus a monoid homomorphism is nothing more than a functor between one-object categories.

Slogan 4.2.1.2.

> “ *A monoid is a category $\mathcal{G}$ with one object. A monoid homomorphism is just a functor between one-object categories.* ”

We formalize this as the following theorem.

Theorem 4.2.1.3. *There is a functor $i\colon$ **Mon** $\to$ **Cat** with the following properties:*

- *for every monoid $\mathcal{M} \in \mathrm{Ob}(\mathbf{Mon})$, the category $i(\mathcal{M}) \in \mathrm{Ob}(\mathbf{Cat})$ itself has exactly one object,*
$$|\mathrm{Ob}(i(\mathcal{M}))| = 1$$

- *for every pair of monoids* $\mathcal{M}, \mathcal{M}' \in \mathrm{Ob}(\mathbf{Mon})$ *the function*

$$\mathrm{Hom}_{\mathbf{Mon}}(\mathcal{M}, \mathcal{M}') \xrightarrow{\cong} \mathrm{Hom}_{\mathbf{Cat}}(i(\mathcal{M}), i(\mathcal{M}')),$$

  induced by the functor i*, is a bijection.*

Proof. This is basically the content of the preceding paragraphs. The functor i sends a monoid to the corresponding category with one object and i sends a monoid homomorphism to the corresponding functor; it is not hard to check that i preserves identities and compositions.

□

Theorem 4.2.1.3 situates the theory of monoids very nicely within the world of categories. But we have other ways of thinking about monoids, namely their actions on sets. As such it would greatly strengthen the story if we could subsume monoid actions within category theory also, and we can.

Each monoid action as a set-valued functor Recall from Definition 3.1.2.1 that if $(M, e, \star)$ is a monoid, an action consists of a set S and a function $\circlearrowleft\colon M \times S \to S$ such that $e \circlearrowleft s = s$ and $m_0 \circlearrowleft (m_1 \circlearrowleft s) = (m_0 \star m_1) \circlearrowleft s$ for all $s \in S$. How might we relate the notion of monoid actions to the notion of functors? One idea is to try asking what a functor $F\colon \mathcal{M} \to \mathbf{Set}$ is; this idea will work.

Since $\mathcal{M}$ has only one object, we obtain one set, $S := F(\blacktriangle) \in \mathrm{Ob}(\mathbf{Set})$. We also obtain a function $\mathrm{Hom}_F\colon \mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle) \to \mathrm{Hom}_{\mathbf{Set}}(F(\blacktriangle), F(\blacktriangle))$, or more concisely, a function

$$H_F\colon M \to \mathrm{Hom}_{\mathbf{Set}}(S, S).$$

By currying (see Proposition 2.7.2.3), this is the same as a function $\circlearrowleft\colon M \times S \to S$. The rule that $e \circlearrowleft s = s$ becomes the rule that functors preserve identities, $\mathrm{Hom}_F(\mathrm{id}_\blacktriangle) = \mathrm{id}_S$. The other rule is equivalent to the composition formula for functors.

4.2.1.4 Groups as categories

A group is just a monoid $(M, e, \star)$ in which every element $m \in M$ is invertible, meaning there exists some $m' \in M$ with $m \star m' = e = m' \star m$. If a monoid is the same thing as a category $\mathcal{M}$ with one object, then a group must be a category with one object and with an additional property having to do with invertibility. The elements of M are the morphisms of the category $\mathcal{M}$, so we need a notion of invertibility for morphisms. Luckily we have such a notion already, namely isomorphism. We have the following:

Slogan 4.2.1.5.

> “ *A group is a category* $\mathcal{G}$ *with one object, such that every morphism in* $\mathcal{G}$ *is an isomorphism. A group homomorphism is just a functor between such categories.* ”

Theorem 4.2.1.6. *There is a functor* $i\colon \mathbf{Grp} \to \mathbf{Cat}$ *with the following properties:*

- *for every group* $\mathcal{G} \in \mathrm{Ob}(\mathbf{Grp})$*, the category* $i(\mathcal{G}) \in \mathrm{Ob}(\mathbf{Cat})$ *itself has exactly one object, and every morphism* m *in* $i(\mathcal{G})$ *is an isomorphism; and*

- *for every pair of groups $\mathcal{G}, \mathcal{G}' \in \mathrm{Ob}(\mathbf{Grp})$ the function*

$$\mathrm{Hom}_{\mathbf{Grp}}(\mathcal{G}, \mathcal{G}') \xrightarrow{\cong} \mathrm{Hom}_{\mathbf{Cat}}(i(\mathcal{G}), i(\mathcal{G}')),$$

  induced by the functor i, is a bijection.

Just as with monoids, an action of some group $(G, e, \star)$ on a set $S \in \mathrm{Ob}(\mathbf{Set})$ is the same thing as a functor $\mathcal{G} \to \mathbf{Set}$ sending the unique object of $\mathcal{G}$ to the set S.

4.2.1.7 Monoid and group stationed at each object in a category

If a monoid is just a category with one object, we can locate monoids in any category $\mathcal{C}$ by narrowing our gaze to one object in $\mathcal{C}$. Similarly for groups.

Example 4.2.1.8 (Endomorphism monoid). Let $\mathcal{C}$ be a category and $x \in \mathrm{Ob}(\mathcal{C})$ an object. Let $M = \mathrm{Hom}_{\mathcal{C}}(x, x)$. Note that for any two elements $f, g \in M$ we have $f \circ g \colon x \to x$ in M. Let $\mathcal{M} = (M, \mathrm{id}_x, \circ)$. It is easy to check that $\mathcal{M}$ is a monoid; it is called the *endomorphism monoid of x in $\mathcal{C}$.*

Example 4.2.1.9 (Automorphism group). Let $\mathcal{C}$ be a category and $x \in \mathrm{Ob}(\mathcal{C})$ an object. Let $G = \{f \colon x \to x \mid f \text{ is an isomorphism}\}$. Let $\mathcal{G} = (G, \mathrm{id}_x, \circ)$. It is easy to check that $\mathcal{G}$ is a group; it is called the *automorphism group of x in $\mathcal{C}$.*

Exercise 4.2.1.10. Let $S = \{1, 2, 3, 4\} \in \mathrm{Ob}(\mathbf{Set})$.

a.) What is the automorphism group of S in **Set**, and how many elements does this group have?

b.) What is the endomorphism monoid of S in **Set**, and how many elements does this monoid have?

c.) Recall from Example 4.1.2.3 that every group has an underlying monoid $U(G)$; is the endomorphism monoid of S the underlying monoid of the automorphism group of S?

◊

Exercise 4.2.1.11. Consider the graph G depicted below.

$$\begin{tikzcd}
\overset{1}{\bullet} \arrow[r, bend left, "12"] \arrow[d, bend left, "13"] & \overset{2}{\bullet} \arrow[l, bend left, "21"] \arrow[d, bend left, "24"] \\
\overset{3}{\bullet} \arrow[u, bend left, "31"] \arrow[r, bend left, "34"] & \overset{4}{\bullet} \arrow[l, bend left, "43"] \arrow[u, bend left, "42"]
\end{tikzcd}$$

What is its group of automorphisms? Hint: every automorphism of G will induce an automorphism of the set $\{1, 2, 3, 4\}$; which ones will preserve the arrows? ◊

4.2.1.12 Preorders as categories

A preorder $(X, \leqslant)$ consists of a set X and a binary relation $\leqslant$ that is reflexive and transitive. We can make from $(X, \leqslant) \in \mathrm{Ob}(\mathbf{PrO})$ a category $\mathcal{X} \in \mathrm{Ob}(\mathbf{Cat})$ as follows. Define $\mathrm{Ob}(\mathcal{X}) = X$ and for every two objects $x, y \in X$ define

$$\mathrm{Hom}_{\mathcal{X}}(x, y) = \begin{cases} \{\text{“}x \leqslant y\text{”}\} & \text{if } x \leqslant y \\ \varnothing & \text{if } x \nleqslant y \end{cases}$$

To clarify: if $x \leqslant y$, we assign $\mathrm{Hom}_{\mathcal{X}}(x, y)$ to be the set containing only one element, namely the string "$x \leqslant y$".[6] If (x, y) is not in relation $\leqslant$, then we assign $\mathrm{Hom}_{\mathcal{X}}(x, y)$ to be the empty set. The composition formula

$$\circ\colon \mathrm{Hom}_{\mathcal{X}}(x, y) \times \mathrm{Hom}_{\mathcal{X}}(y, z) \to \mathrm{Hom}_{\mathcal{X}}(x, z) \tag{4.6}$$

is completely determined because either one of two possibilities occurs. One possibility is that the left-hand side is empty (if either $x \nleqslant y$ or $y \nleqslant z$; in this case there is a unique function $\circ$ as in (4.6). The other possibility is that the left-hand side is not empty in case $x \leqslant y$ and $y \leqslant$, which implies $x \leqslant z$, so the right-hand side has exactly one element "$x \leqslant z$" in which case again there is a unique function $\circ$ as in (4.6).

On the other hand, if $\mathcal{C}$ is a category having the property that for every pair of objects $x, y \in \mathrm{Ob}(\mathcal{C})$, the set $\mathrm{Hom}_{\mathcal{C}}(x, y)$ is either empty or has one element, then we can form a preorder out of $\mathcal{C}$. Namely, take $X = \mathrm{Ob}(\mathcal{C})$ and say $x \leqslant y$ if there exists a morphism $x \to y$ in $\mathcal{C}$.

Exercise 4.2.1.13. We have seen that a preorder can be considered as a category $\mathcal{P}$. Recall from Definition 3.4.1.1 that a partial order is a preorder with an additional property. Phrase the defining property for partial orders in terms of isomorphisms in the category $\mathcal{P}$. ◊

Exercise 4.2.1.14. Suppose that $\mathcal{C}$ is a preorder (considered as a category). Let $x, y \in \mathrm{Ob}(\mathcal{C})$ be objects such that $x \leqslant y$ and $y \leqslant x$. Prove that there is an isomorphism $x \to y$ in $\mathcal{C}$. ◊

Example 4.2.1.15. The olog from Example 3.4.1.3 depicted a partial order, say $\mathcal{P}$. In it we have

$$\mathrm{Hom}_{\mathcal{P}}(\ulcorner\text{a diamond}\urcorner, \ulcorner\text{a red card}\urcorner) = \{\text{is}\}$$

and we have

$$\mathrm{Hom}_{\mathcal{P}}(\ulcorner\text{a black queen}\urcorner, \ulcorner\text{a card}\urcorner) \cong \{\text{is} \circ \text{is}\};$$

Both of these sets contain exactly one element, the name is not important. The set $\mathrm{Hom}_{\mathcal{P}}(\ulcorner\text{a 4}\urcorner, \ulcorner\text{a 4 of diamonds}\urcorner) = \varnothing$.

Exercise 4.2.1.16. Every linear order is a partial order with a special property. Can you phrase this property in terms of hom-sets? ◊

Proposition 4.2.1.17. *There is a functor* $i\colon \mathbf{PrO} \to \mathbf{Cat}$ *with the following properties for every preorder* $(X, \leqslant)$*:*

1. *the category* $\mathcal{X} := i(X, \leqslant)$ *has objects* $\mathrm{Ob}(\mathcal{X}) = X$*; and*
2. *for each pair of elements* $x, x' \in \mathrm{Ob}(\mathcal{X})$ *the set* $\mathrm{Hom}_{\mathcal{X}}(x, x')$ *has at most one element.*

Moreover, any category with property 2 is in the image of the functor i*.*

Proof. To specify a functor $i\colon \mathbf{PrO} \to \mathbf{Cat}$, we need to say what it does on objects and on morphisms. To an object $(X, \leqslant)$ in $\mathbf{PrO}$, we assign the category $\mathcal{X}$ with objects X and a unique morphism from $x \to x'$ if $x \leqslant x'$; this was discussed at the top of Section 4.2.1.12. To a morphism $f\colon (X, \leqslant_X) \to (Y, \leqslant_Y)$ of preorders, we must assign a functor $i(f)\colon \mathcal{X} \to \mathcal{Y}$. Again, to specify a functor we need to say what it does on objects and

[6]The name of this morphism is completely unimportant. What matters is that $\mathrm{Hom}_{\mathcal{X}}(x, y)$ has exactly one element iff $x \leqslant y$.

morphisms of $\mathcal{X}$. To an object $x \in \mathrm{Ob}(\mathcal{X}) = X$, we assign the object $f(x) \in Y = \mathrm{Ob}(\mathcal{Y})$. Given a morphism $f\colon x \to x'$ in $\mathcal{X}$, we know that $x \leqslant x'$ so by Definition 3.4.4.1 we have that $f(x) \leqslant f(x')$, and we assign to f the unique morphism $f(x) \to f(x')$ in $\mathcal{Y}$. To check that the rules of functors (preservation of identities and composition) are obeyed is routine.

□

Slogan 4.2.1.18.

> " *A preorder is a category in which every hom-set has either 0 elements or 1 element. A preorder morphism is just a functor between such categories.* "

Exercise 4.2.1.19. Recall the functor $P\colon \mathbf{PrO} \to \mathbf{Grph}$ from Proposition 4.1.2.8, the functors $F\colon \mathbf{Grph} \to \mathbf{Cat}$ and $U\colon \mathbf{Cat} \to \mathbf{Grph}$ from Example 4.1.2.33, and the functor $i\colon \mathbf{PrO} \to \mathbf{Cat}$ from Proposition 4.2.1.17.

a.) Do either of the following diagrams of categories commute?

$$\begin{tikzcd}
\mathbf{PrO} \arrow[rr, "P"] \arrow[dr, "i"'] & {?} & \mathbf{Grph} \arrow[dl, "F"] \\
& \mathbf{Cat} &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{PrO} \arrow[rr, "P"] \arrow[dr, "i"'] & {?} & \mathbf{Grph} \\
& \mathbf{Cat} \arrow[ur, "U"'] &
\end{tikzcd}$$

b.) We also had a functor $\mathbf{Grph} \to \mathbf{PrO}$. Does the following diagram of categories commute?

$$\begin{tikzcd}
\mathbf{Grph} \arrow[rr] \arrow[dr, "F"'] & {?} & \mathbf{PrO} \arrow[dl, "i"] \\
& \mathbf{Cat} &
\end{tikzcd}$$

◊

4.2.1.20 Graphs as functors

Let $\mathcal{C}$ denote the category depicted below

$$\mathbf{GrIn} := \boxed{\begin{tikzcd} \overset{Ar}{\bullet} \arrow[r, shift left, "src"] \arrow[r, shift right, "tgt"'] & \overset{Ve}{\bullet} \end{tikzcd}} \tag{4.7}$$

Then a functor $G\colon \mathbf{GrIn} \to \mathbf{Set}$ is the same thing as two sets $G(Ar), G(Ve)$ and two functions $G(src)\colon G(Ar) \to G(Ve)$ and $G(tgt)\colon G(Ar) \to G(Ve)$. This is precisely what is needed for a graph; see Definition 3.3.1.1. We call **GrIn** the *graph indexing category*.

Exercise 4.2.1.21. Consider the terminal category, $\underline{1}$, also known as the discrete category on one element (see Exercise 4.1.2.37). Let **GrIn** be as in (4.7) and consider the functor $i_0\colon \underline{1} \to \mathbf{GrIn}$ sending the object of $\underline{1}$ to the object $V \in \mathrm{Ob}(\mathbf{GrIn})$. If $G\colon \mathbf{GrIn} \to \mathbf{Set}$ is a graph, what is the composite $G \circ i_0$? It consists of only one set; what set is it? For example, what set is it when G is the graph from Example 3.3.3.3. ◊

If a graph is a functor **GrIn** → **Set**, what is a graph homomorphism? We will see later in Example 4.3.1.17 that graph homomorphisms are homomorphisms between functors, which are called natural transformations. (Natural transformations are the highest-"level" structure that occurs in ordinary category theory.)

Example 4.2.1.22. Let $\mathcal{D}$ be the category depicted below

$$\mathcal{D} := \boxed{\begin{tikzcd} \overset{A}{\bullet} \arrow[loop left, "\rho"] \arrow[r, shift left, "src"] \arrow[r, shift right, "tgt"'] & \overset{V}{\bullet} \end{tikzcd}} \tag{4.8}$$

with the following composition formula:

$$\rho \circ \rho = \mathrm{id}_A; \qquad src \circ \rho = tgt; \qquad \text{and} \qquad tgt \circ \rho = src.$$

The idea here is that the morphism $\rho\colon A \to A$ reverses arrows. The PED $\rho \circ \rho = \mathrm{id}_A$ forces the fact that the reverse of the reverse of an arrow yields the original arrow. The PEDs $src \circ \rho = tgt$ and $tgt \circ \rho = src$ force the fact that when we reverse an arrow, its source and target switch roles.

This category $\mathcal{D}$ is the *symmetric graph indexing category.* Just like any graph can be understood as a functor **GrIn** → **Set**, where **GrIn** is the graph indexing category displayed in (4.7), any symmetric graph can be understood as a functor $\mathcal{D}$ → **Set**, where $\mathcal{D}$ is the category drawn above. Given a functor $G\colon \mathcal{D}$ → **Set**, we will have a set of arrows, a set of vertices, a source operation, a target operation, and a "reverse direction" operation that all behave as expected.

It is customary to draw the connections in a symmetric graph as line segments rather than arrows between vertices. However, a better heuristic is to think that each connection between vertices consists of two arrows, one pointing in each direction.

Slogan 4.2.1.23.

" *In a symmetric graph, every arrow has an equal and opposite arrow.* "

Exercise 4.2.1.24. Which of the following graphs are symmetric:

a.) The graph G from (3.4)?

b.) The graph G from Exercise 3.3.1.10?

c.) The graph G' from (3.7)?

d.) The graph $\mathcal{L}oop$ from (3.17), i.e. the graph having exactly one vertex and one arrow?

e.) The graph G from Exercise 4.2.1.11?

◊

Exercise 4.2.1.25. Let **GrIn** be the graph indexing category shown in (4.7) and let $\mathcal{D}$ be the symmetric graph indexing category displayed in (4.8).

a.) How many functors are there of the form **GrIn** → $\mathcal{D}$?

b.) Is one more "reasonable" than the others?

c.) Choose the one that seems most reasonable and call it $i\colon$ **GrIn** → $\mathcal{D}$. If a symmetric graph is a functor $S\colon \mathcal{D}$ → **Set**, you can compose with i to get a functor $S \circ i\colon$ **GrIn** → **Set**. This is a graph; what graph is it? What has changed?

◊

4.2.2 Database schemas present categories

Recall from Definition 3.5.2.6 that a database schema (or schema, for short) consists of a graph together with a certain kind of equivalence relation on its paths. In Section 4.4.1 we will define a category **Sch** that has schemas as objects and appropriately modified graph homomorphisms as morphisms. In Section 4.4.2 we prove that the category of schemas is equivalent (in the sense of Definition 4.3.4.1) to the category of categories,

$$\mathbf{Sch} \simeq \mathbf{Cat}.$$

The difference between schemas and categories is like the difference between monoid presentations, given by generators and relations as in Definition 3.1.1.17, and the monoids themselves. The same monoid has (infinitely) many different presentations, and so it is for categories: many different schemas can *present* the same category. Computer scientists may think of the schema as *syntax* and the category it presents as the corresponding *semantics*. A schema is a compact form, and can be specified in finite space and time while generating something infinite.

Slogan 4.2.2.1.

“ *A database schema is a category presentation.* ”

We will formally show in Section 4.4.2 how to turn a schema into a category (the category it *presents*). For now, it seems pedagogically better not to be so formal, because the idea is fairly straightforward. Suppose given a schema $\mathcal{S}$, which consists of a graph $G = (V, A, src, tgt)$ equipped with a congruence $\sim$ (see Definition 3.5.2.3). It presents a category $\mathcal{C}$ defined as follows. The set of objects in $\mathcal{C}$ is defined to be the vertices V; the set of morphisms in $\mathcal{C}$ is defined to be the quotient $\text{Paths}(G)/\sim$; and the composition law is concatenation of paths. The path equivalences making up $\sim$ become commutative diagrams in $\mathcal{C}$.

Example 4.2.2.2. The schema $\mathcal{L}oop$, depicted below, has no path equivalence declarations. As a graph it has one vertex and one arrow.

$$\mathcal{L}oop := \boxed{\begin{tikzcd} \overset{s}{\bullet} \arrow[loop above, "f"] \end{tikzcd}}$$

The category it generates, however, is the free monoid on one generator, $\mathbb{N}$. It has one object $\blacktriangle$ but a morphism $f^n \colon \blacktriangle \to \blacktriangle$ for every natural number $n \in \mathbb{N}$, thought of as “how many times to go around the loop f”. Clearly, the schema is more compact that the infinite category it generates.

Exercise 4.2.2.3. Consider the olog from Exercise 3.5.2.18, which says that for any father x, his first child's father is x. It is redrawn below as a schema $\mathcal{S}$, and we include the desired path equivalence declaration, $F\ c\ f = F$,

$$\begin{tikzcd} \overset{F}{\bullet} \arrow[r, "c"] & \overset{C}{\bullet} \arrow[l, bend left, "f"] \end{tikzcd}$$

How many morphisms are there (total) in the category generated by $\mathcal{S}$? ◊

Exercise 4.2.2.4. Suppose that G is a graph and that $\mathcal{G}$ is the schema generated by G with no PEDs. What is the relationship between the category generated by $\mathcal{G}$ and the free category $F(G) \in \text{Ob}(\mathbf{Cat})$ as defined in Example 4.1.2.30? ◊

4.2.2.5 Instances on a schema $\mathcal{C}$

If schemas are like categories, what are instances? Recall that an instance I on a schema $\mathcal{S} = (G, \simeq)$ assigns to each vertex v in G a set of rows say $I(v) \in \mathrm{Ob}(\mathbf{Set})$. And to every arrow $a\colon v \to v'$ in G the instance assigns a function $I(a)\colon I(v) \to I(v')$. The rule is that given two equivalent paths, their compositions must give the same function. Concisely, an instance is a functor $I\colon \mathcal{S} \to \mathbf{Set}$.

Example 4.2.2.6. We have now seen that a monoid is just a category $\mathcal{M}$ with one object and that a monoid action is a functor $\mathcal{M} \to \mathbf{Set}$. Under our understanding of database schemas as categories, $\mathcal{M}$ is a schema and so an action becomes an instance of that schema. The monoid action table from Example ex:action table was simply a manifestation of the database instance according to the Rules 3.5.2.8.

Exercise 4.2.2.7. In Section 4.2.1.20 we discuss how each graph is a functor $\mathbf{GrIn} \to \mathbf{Set}$ for the graph indexing category depicted below:

$$\mathbf{GrIn} := \boxed{\begin{tikzcd} \overset{Ar}{\bullet} \arrow[r, shift left, "src"] \arrow[r, shift right, "tgt"'] & \overset{Ve}{\bullet} \end{tikzcd}}$$

But now we know that if a graph is a set-valued functor then we can consider **GrIn** as a database schema.

a.) How many tables, and how many columns of each should there be (if unsure, consult Rules 3.5.2.8)?

b.) Write out the table view of graph G from Example 3.3.3.3.

◊

4.2.3 Spaces

Category theory was invented for use in algebraic topology, and in particular to discuss natural transformations between certain functors. We will get to natural transformations more formally in Section 4.3. For now, they are ways of relating functors. In the original use, Eilenberg and Mac Lane were interested in functors that connect topological spaces (shapes like spheres, etc.) to algebraic systems (groups, etc.)

For example, there is a functor that assigns to each space X its group $\pi_1(X)$ of round-trip voyages (starting and ending at some chosen point $x \in X$), modulo some equivalence relation. There is another functor that assigns to every space its group $H_1(X, \mathbb{Z})$ of ways to drop some (positive or negative) number of circles on X. These two functors are related, but they are not equal.

There is a relationship between the functor π_1 and the functor H_1. For example when X is the figure-8 space (two circles joined at a point) the group $\pi_1(X)$ is much bigger than the group $H_1(X)$. Indeed $\pi_1(X)$ includes information about the order and direction of loops traveled; whereas the group $H_1(X, \mathbb{Z})$ includes only information about how many times one goes around each loop. However, there is a natural transformation of functors $\pi_1(-) \to H_1(-, \mathbb{Z})$, called the Hurewicz transformation, which "forgets" the extra information and thus yields a simplification.

Example 4.2.3.1. Given a set X, recall that $\mathbb{P}(X)$ denotes the set of subsets of X. A *topology* on X is a choice of which subsets $U \in \mathbb{P}(X)$ will be called *open sets*. The union of any number of open sets must be considered to be an open set, and the intersection of any finite number of open sets must be considered open. One could say succinctly that a topology on X is a sub-order $\text{Open}(X) \subseteq \mathbb{P}(X)$ that is closed under taking finite meets and infinite joins.

A *topological space* is a pair $(X, \text{Open}(X))$, where X is a set and $\text{Open}(X)$ is a topology on X. The elements of the set X are called *points*. A *morphism of topological spaces* (also called a *continuous map*) is a function $f\colon X \to Y$ such that for every $V \in \text{Open}(Y)$ the preimage $f^{-1}(V) \in \mathbb{P}(X)$ is actually in $\text{Open}(X)$. That is, such that there exists a dashed arrow making the diagram below commute:

$$\begin{tikzcd}
\text{Open}(Y) \arrow[r, dashed] \arrow[d] & \text{Open}(X) \arrow[d] \\
\mathbb{P}(Y) \arrow[r, "f^{-1}"'] & \mathbb{P}(X).
\end{tikzcd}$$

The *category of topological spaces*, denoted **Top**, is the category having objects and morphisms as above.

Exercise 4.2.3.2.

a.) Explain how "looking at points" gives a functor $\textbf{Top} \to \textbf{Set}$.

b.) Does "looking at open sets" give a functor $\textbf{Top} \to \textbf{PrO}$?

◊

Example 4.2.3.3 (Continuous dynamical systems). The set $\mathbb{R}$ can be given a topology in a standard way.[7] But $(\mathbb{R}, 0, +)$ is also a monoid. Moreover, for every $x \in \mathbb{R}$ the monoid operation $+\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is continuous. [8] So we say that $\mathcal{R} := (\mathbb{R}, 0, +)$ is a *topological monoid.*

Recall from Section 4.2.1.1 that a monoid action is a functor $\mathcal{M} \to \textbf{Set}$, where $\mathcal{M}$ is a monoid. Instead imagine a functor $a\colon \mathcal{R} \to \textbf{Top}$? Since $\mathcal{R}$ is a category with one object, this amounts to an object $X \in \text{Ob}(\textbf{Top})$, a space. And to every real number $t \in \mathbb{R}$ we obtain a continuous map $a(t)\colon X \to X$. If we consider X as the set of states of some system and $\mathbb{R}$ as the time line, we have captured what is called a *continuous dynamical system.*

Example 4.2.3.4. Recall (see [Axl]) that a *real vector space* is a set X, elements of which are called *vectors*, which is closed under addition and scalar multiplication. For example $\mathbb{R}^3$ is a vector space. A *linear transformation from X to Y* is a function $f\colon X \to Y$ that appropriately preserves addition and scalar multiplication. The *category of real vector spaces*, denoted $\textbf{Vect}_{\mathbb{R}}$, has as objects the real vector spaces and as morphisms the linear transformations.

There is a functor $\textbf{Vect}_{\mathbb{R}} \to \textbf{Grp}$ sending a vector space to its underlying group of vectors, where the group operation is addition of vectors and the group identity is the 0-vector.

Exercise 4.2.3.5. Every vector space has vector subspaces, ordered by inclusion (the origin is inside of any line which is inside of certain planes, etc., and all are inside of the whole space V). If you know about this topic, answer the following questions.

[7] The topology is given by saying that $U \subseteq \mathbb{R}$ is open iff for every $x \in U$ there exists $\epsilon > 0$ such that $\{y \in \mathbb{R} \mid |y - x| < \epsilon\} \subseteq U\}$. One says, "$U \subseteq \mathbb{R}$ is open if every point in U has an epsilon-neighborhood fully contained in U".

[8] The topology on $\mathbb{R} \times \mathbb{R}$ is similar; a subset $U \subseteq \mathbb{R} \times \mathbb{R}$ is open if every point $x \in U$ has an epsilon-neighborhood (a disk around x of some positive radius) fully contained in U.

a.) Does a linear transformation $V \to V'$ induce a morphism of these orders? In other words, is there a functor $\mathbf{Vect}_{\mathbb{R}} \to \mathbf{PrO}$?

b.) Would you guess that there is a nice functor $\mathbf{Vect}_{\mathbb{R}} \to \mathbf{Top}$? By a "nice functor" I mean one that doesn't make people roll their eyes (for example, there is a functor $\mathbf{Vect}_{\mathbb{R}} \to \mathbf{Top}$ that sends every vector space to the empty space, and that's not really a "nice" one. If someone asked for a functor $\mathbf{Vect}_{\mathbb{R}} \to \mathbf{Top}$ for their birthday, this functor would make them sad. We're looking for a functor $\mathbf{Vect}_{\mathbb{R}} \to \mathbf{Top}$ that would make them happy.)

$\lozenge$

4.2.3.6 Groupoids

Groupoids are like groups except a groupoid can have more than one object.

Definition 4.2.3.7. A *groupoid* is a category $\mathcal{C}$ such that every morphism is an isomorphism. If $\mathcal{C}$ and $\mathcal{D}$ are groupoids, a *morphism of groupoids*, denoted $F: \mathcal{C} \to \mathcal{D}$, is simply a functor. The category of groupoids is denoted **Grpd**.

Example 4.2.3.8. There is a functor $\mathbf{Grpd} \to \mathbf{Cat}$, sending a groupoid to its underlying category. There is also a functor $\mathbf{Grp} \to \mathbf{Grpd}$ sending a group to "itself as a groupoid with one object."

Application 4.2.3.9. Let M be a material in some original state s_0.[9] Construct a category $\mathcal{S}_M$ whose objects are the states of M, e.g. by pulling on M in different ways, or by heating it up, etc. we obtain such states. Include a morphism from state s to state s' if there exists a physical transformation from s to s'. Physical transformations can be performed one after another, so we can compose morphisms, and perhaps we can agree this composition is associative. Note that there exists a morphism $i_s: s_0 \to s$ for any s. Note also that this category is a preorder because there either exists a physical transformation or there does not. [10]

The elastic deformation region of the material is the set of states s such that there exists a morphism $s \to s_0$, because any such morphism will be the inverse of $i_s: s_0 \to s$. A transformation is irreversible if there is no transformation back. If s_1 is not in the elastic deformation region, we can (inventing a term) still talk about the region that is "elastically-equivalent" to s_1. It is all the objects in $\mathcal{S}_M$ that are isomorphic to s_1. If we consider only elastic equivalences, we are looking at a groupoid sitting inside the larger category $\mathcal{S}_M$.

$\lozenge\lozenge$

Example 4.2.3.10. Alan Weinstein explains groupoids in terms of tiling patterns on a bathroom floor, see [WeA].

Example 4.2.3.11. Let $I = \{x \in \mathbb{R} \mid 0 \leqslant x \leqslant 1\}$ denote the unit interval. It can be given a topology in a standard way, as a subset of $\mathbb{R}$ (see Example 4.2.3.3)

For any space X, a *path in* X is a continuous map $I \to X$. Two paths are called *homotopic* if one can be continuously deformed to the other, where the deformation

[9]This example may be a bit crude, in accordance with the crudeness of my understanding of materials science.

[10]Someone may choose to beef this category up to include the set of physical processes between states as the hom-set. This gives a category that is not a preorder. But there would be a functor from their category to ours.

occurs completely within X. [11] One can prove that being homotopic is an equivalence relation on paths.

Paths in X can be composed, one after the other, and the composition is associative (up to homotopy). Moreover, for any point $x \in X$ there is a trivial path (that stays at x). Finally every path is invertible (by traversing it backwards) up to homotopy.

This all means that to any space $X \in \mathrm{Ob}(\mathbf{Top})$ we can associate a groupoid, called the *fundamental groupoid of* X and denoted $\Pi_1(X) \in \mathrm{Ob}(\mathbf{Grpd})$. The objects of $\Pi_1(X)$ are the points of X; the morphisms in $\Pi_1(X)$ are the paths in X (up to homotopy). A continuous map $f\colon X \to Y$ can be composed with any path $I \to X$ to give a path $I \to Y$ and this preserves homotopy. So in fact $\Pi_1\colon \mathbf{Top} \to \mathbf{Grpd}$ is a functor.

Exercise 4.2.3.12. Let T denote the surface of a donut, i.e. a torus. Choose two points $p, q \in T$. Since $\Pi_1(T)$ is a groupoid, it is also a category. What would the hom-set $\mathrm{Hom}_{\Pi_1(T)}(p, q)$ represent? ◊

Exercise 4.2.3.13. Let $U \subseteq \mathbb{R}^2$ be an open subset of the plane, and let F be an irrotational vector field on U (i.e. one with $\mathrm{curl}(F) = 0$). Following Exercise 4.1.1.15, we have a category $\mathcal{C}_F$. If two curves C, C' in U are homotopic then they have the same line integral, $\int_C F = \int_{C'} F$.

We also have a category $\Pi_1 U$, given by the fundamental groupoid, as in Example 4.2.3.11. Both categories have the same objects, $\mathrm{Ob}(\mathcal{C}_F) = |U| = \mathrm{Ob}(\Pi_1 U)$, the set of points in U.

a.) Is there a functor $\mathcal{C}_F \to \Pi_1 U$ or a functor $\Pi_1 U \to \mathcal{C}_F$ that is identity on the underlying objects?

b.) What is $\mathcal{C}_F$ if F is a conservative vector field?

◊

Exercise 4.2.3.14. Consider the set A of all (well-formed) arithmetic expressions in the symbols $\{0, \ldots, 9, +, -, *, (,)\}$. For example, here are some elements of A:

$$52, \qquad 52 - 7, \qquad 50 + 3 * (6 - 2).$$

We can say that an equivalence between two arithmetic expressions is a justification that they give the same "final answer", e.g. $52 + 60$ is equivalent to $10 * (5 + 6) + (2 + 0)$, which is equivalent to $10 * 11 + 2$. I've basically described a groupoid. What are its objects and what are its morphisms? ◊

4.2.4 Logic, set theory, and computer science

4.2.4.1 The category of propositions

Given a domain of discourse, a logical proposition is a statement that is evalued in any model of that domain as either true or "not always true". For example, in the domain of real numbers we might have the proposition

For all real numbers $x \in \mathbb{R}$ there exists a real number $y \in \mathbb{R}$ such that $y > 3x$.

[11] Let $I^2 = \{(x, y) \in \mathbb{R}^2 \mid 0 \leqslant x \leqslant 1 \text{ and } 0 \leqslant y \leqslant 1\}$ denote the square. There are two inclusions $i_0, i_1\colon I \to S$ that put the interval inside the square at the left and right sides. Two paths $f_0, f_1\colon I \to X$ are homotopic if there exists a continuous map $f\colon I \times I \to X$ such that $f_0 = f \circ i_0$ and $f_1 = f \circ i_1$,

$$\begin{tikzcd} I \arrow[r, shift left, "i_0"] \arrow[r, shift right, "i_1"'] & I \times I \arrow[r, "f"] & X \end{tikzcd}$$

We say that one logical proposition P *implies* another proposition Q, denoted $P \Rightarrow Q$ if, for every model in which P is true, so is Q. There is a category **Prop** whose objects are logical propositions and whose morphisms are proofs that one statement implies another. Crudely, one might say that B *holds at least as often as* A if there is a morphism $A \to B$ (meaning whenever A holds, so does B). So the proposition "$x \neq x$" holds very seldom and "$x = x$" always very often.

Example 4.2.4.2. We can repeat this idea for non-mathematical statements. Take all possible statements that are verifiable by experiment as objects of a category. Given two such statements, it may be that one implies the other (e.g. "if the speed of light is fixed then there are relativistic effects"). Every statement implies itself (identity) and implication is transitive, so we have a category.

Let's consider differences in proofs to be irrelevant, so the category **Prop** becomes a preorder: either A implies B or it does not. Then it makes sense to discuss meets and joins. It turns out that meets are "and's" and joins are "or's". That is, given propositions A, B the meet $A \wedge B$ is defined to be a proposition that holds as often as possible subject to the constraint that it implies both A and B; the proposition "A holds and B holds" fits the bill. Similarly, the join $A \vee B$ is given by "A holds or B holds".

Exercise 4.2.4.3. Consider the set of possible laws (most likely an infinite set) that can be dictated to hold throughout a jurisdiction. Consider each law as a proposition ("such and such is (dictated to be) the case"), i.e as an object of our preorder **Prop**. Given a jurisdiction V, and a set of laws $\{\ell_1, \ell_2, \ldots, \ell_n\}$ that are dictated to hold throughout V, we take their meet $L(V) := \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n$ and consider it to be the single law of the land V. Suppose that V is a jurisdiction and U is a sub-jurisdiction (e.g. U is a county and V is a state); write $U \leqslant V$. Then clearly any law dictated by the large jurisdiction (the state) must also hold throughout the small jurisdiction (the county).

a.) What is the relation in **Prop** between $L(U)$ and $L(V)$?

b.) Consider the preorder J on jurisdictions given by $\leqslant$ as above. Is "the law of the land" a morphism of preorders $J \to$ **Prop**? To be a bit more high-brow, considering both J and **Prop** to be categories (by Proposition 4.2.1.17), we have a function $L\colon \mathrm{Ob}(J) \to \mathrm{Ob}(\mathbf{Prop})$; this question is asking whether L extends to a functor $J \to$ **Prop**.[12]

◊

Exercise 4.2.4.4. Take again the preorder J of jurisdictions from Exercise 4.2.4.3 and the idea that laws are propositions. But this time, let $R(V)$ be the set of all possible laws (not just those dictated to hold) that are in actuality being respected, i.e. followed, by all people in V. This assigns to each jurisdiction a set.

a.) Since preorders can be considered categories, does our "the set of respected laws" function $R\colon \mathrm{Ob}(J) \to \mathrm{Ob}(\mathbf{Set})$ extend to a functor $J \to$ **Set**?

b.) What about if instead we take the meet of all these laws and assign to each jurisdiction the maximal law respected throughout. Does this assignment $\mathrm{Ob}(J) \to \mathrm{Ob}(\mathbf{Prop})$ extend to a functor $J \to$ **Prop**? [12]

◊

[12]Hint: Exercises 4.2.4.3 and 4.2.4.4 will ask similar yes/no questions and at least one of these is correctly answered "no".

4.2.4.5 A categorical characterization of Set

The category **Set** of sets is fundamental in mathematics, but instead of thinking of it as something given or somehow special, it can be shown to merely be a category with certain properties, each of which can be phrased purely categorically. This was shown by Lawvere [Law]. A very readable account is given in [Le2].

4.2.4.6 Categories in computer science

Computer science makes heavy use of trees, graphs, orders, lists, and monoids. We have seen that all of these are naturally viewed in the context of category theory, though it seems that such facts are rarely mentioned explicitly in computer science textbooks. However, categories are also used explicitly in the theory of programming languages (PL). Researchers in that field attempt to understand the connection between what programs are supposed to do (their denotation) and what they actually cause to occur (their operation). Category theory provides a useful mathematical formalism in which to study this.

The kind of category most often considered by a PL researcher is what is known as a *Cartesian closed category* or *CCC*, which means a category $\mathcal{T}$ that has products (like $A \times B$ in **Set**) and exponential objects (like B^A in **Set**). **Set** is an example of a CCC, but there are others that are more appropriate for actual computation. The objects in a PL person's CCC represent the *types* of the language, types such as `integers, strings, floats`. The morphisms represent computable functions, e.g. `length: strings`$\longrightarrow$`integers`. The products allow one to discuss pairs (a, b) where a is of one type and b is of another type. Exponential objects allow one to consider computable functions as things that can be input to a function (e.g. given any computable function `floats`$\to$`integers` one can consistently multiply its results by 2 and get a new computable function `floats`$\to$`integers`. We will be getting to products in Section 4.5.1.8 and exponential objects in Section 4.3.2.

But category theory did not only offer a language for thinking about programs, it offered an unexpected tool called monads. The above CCC model for types allows researchers only to discuss functions, leading to the notion of functional programming languages; however, not all things that a computer does are functions. For example, reading input and output, changing internal state, etc. are operations that can be performed that ruin the functional-ness of programs. Monads were found in 19?? by Moggi [Mog] to provide a powerful abstraction that opens the doors to such non-functional operations without forcing the developer to leave the category-theoretic garden of eden. We will discuss monads in Section 5.3.

We have also seen in Section 4.2.2 that databases are well captured by the language of categories. We will formalize this in Section 4.4. Throughout the remainder of this book we will continue to use databases to bring clarity to concepts within standard category theory.

4.2.5 Categories applied in science

Categories are being used throughout mathematics to relate various subjects, as well as to draw out the essential structures within these subjects. For example, there is an active research for "categorifying" classical theories like that of knots, links, and braids [Kho]. It is similarly applied in science, to clarify complex subjects. Here are some very brief descriptions of scientific disciplines to which category theory is applied.

Quantum field theory is was categorified by Atiyah [Ati] in the late 1980's, with much success (at least in producing interesting mathematics). In this domain, one takes a category in which an object is a reasonable space, called a manifold, and a morphism is a manifold connecting two manifolds, like a cylinder connects two circles. Such connecting manifolds are called cobordisms, and as such people refer to the category as **Cob**. Topological quantum field theory is the study of functors $\mathbf{Cob} \to \mathbf{Vect}$ that assign a vector space to each manifold and a linear transformation of vector spaces to each cobordism.

Information theory [13] is the study of how to ideally compress messages so that they can be sent quickly and accurately across a noisy channel.[14] Invented in 1948 by Claude Shannon, its main quantity of interest is the number of bits necessary to encode a piece of information. For example, the amount of information in an English sentence can be greatly reduced. The fact that `t`'s are often followed by `h`'s, or that `e`'s are much more common than `z`'s, implies that letters are not being used as efficiently as possible. The amount of bits necessary to encode a message is called its *entropy* and has been linked to the commonly used notion of the same name in physics.

In [BFL], Baez, Fritz, and Leinster show that entropy can be captured quite cleanly using category theory. They make a category `FinProb` whose objects are finite sets equipped with a probability measure, and whose morphisms are probability preserving functions. They characterize *information loss* as a way to assign numbers to such morphisms, subject to certain explicit constraints. They then show that the entropy of an object in `FinProb` is the amount of information lost under the unique map to the singleton set {☺}. This approach explicates (by way of the explicit constraints for information loss functions) the essential idea of Shannon's information theory, allowing it to be generalized to categories other than `FinProb`. Thus Baez and Leinster effectively *categorified* information theory.

Robert Rosen proposed in the 1970s that category theory could play a major role in biology. That story is only now starting to be fleshed out. There is a categorical account of evolution and memory, called *Memory Evolutive Systems* [EV]. There is also a paper [BP2] by Brown and Porter with applications to neuroscience.

4.3 Natural transformations

In this section we conclude our discussion of the **Big 3**, by defining natural transformations. Category theory was originally invented to discuss natural transformations. These were sufficiently conceptually challenging that they required formalization and thus the invention of category theory. If we think of categories as domains (of discourse, interaction, comparability, etc.) and of functors as transformations between different domains, the natural transformations compare different transformations.

Natural transformations can seem a bit abstruse at first, but hopefully some examples and exercises will help.

[13] To me, the subject of "information theory" is badly named. That discipline is devoted to finding ideal compression schemes for messages to be sent quickly and accurately across a noisy channel. It deliberately does not pay any attention to what the messages mean. To my mind this should be called compression theory or redundancy theory. Information is inherently meaningful—that is its purpose—any theory that is unconcerned with the meaning is not really studying information per se. The people who decide on speed limits for roads and highways may care about human health, but a study limited to deciding ideal speed limits should not be called "human health theory".

[14] Despite what was said above, Information theory has been extremely important in a diverse array of fields, including computer science [MacK], but also in neuroscience [Bar], [Lin] and physics [Eve]. I'm not trying to denigrate the field; I am only frustrated with its name.

4.3.1 Definition and examples

Let's begin with an example. There is a functor List: **Set** $\to$ **Set**, which sends a set X to the set $\mathrm{List}(X)$ consisting of all lists whose entries are elements of X. Given a morphism $f\colon X \to Y$, we can transform a list with entries in X into a list with entries in Y by applying f to each (this was worked out in Exercise 4.1.2.20)..

It may seem a strange thing to contemplate, but there is also a functor List $\circ$ List: **Set** $\to$ **Set** that sends a set X to the set of lists of lists in X. If $X = \{a,b,c\}$ then $\mathrm{List}\circ\mathrm{List}(X)$ contains elements like $\big[[a,b],[a,c,a,b,c],[c]\big]$ and $\big[[\,]\big]$ and $\big[[a],[\,],[a,a,a]\big]$. We can *naturally transform* a list of lists into a list by concatenation. In other words, for any set X there is a function $\mu_X\colon \mathrm{List}\circ\mathrm{List}(X) \to \mathrm{List}(X)$ which sends our lists above to $[a,b,a,c,a,b,c,c]$ and $[\,]$ and $[a,a,a,a]$, respectively. In fact, even if we use a function $f\colon X \to Y$ to convert a list of X's into a list of Y's (or a list of lists of X's into a list of lists of Y's), the concatenation "works right". Take a deep breath for the precise statement couched as a slogan.

Slogan 4.3.1.1.

> " *Naturality works like this: Using a function $f\colon X \to Y$ to convert a list of lists of X's into a list of list of Y's and then concatenating to get a simple list of Y's* **does the same thing as** *first concatenating our list of lists of X's into a simple list of X's and then using our function f to convert it into a list of Y's.* "

Let's make this concrete. Let $X = \{a,b,c\}$, let $Y = \{1,2,3\}$, and let $f\colon X \to Y$ assign $f(a) = 1, f(b) = 1, f(c) = 2$. Our naturality condition says the following for any list of lists of X's, in particular for $\big[[a,b],[a,c,a,b,c],[c]\big]$:

$$\begin{tikzcd}
\big[[a,b],[a,c,a,b,c],[c]\big] \arrow[r, mapsto, "\mu_X"] \arrow[d, mapsto, "\mathrm{List}\circ\mathrm{List}(f)"'] & {[a,b,a,c,a,b,c,c]} \arrow[d, mapsto, "\mathrm{List}(f)"] \\
\big[[1,1],[1,2,1,1,2],[2]\big] \arrow[r, mapsto, "\mu_Y"'] & {[1,1,1,2,1,1,2,2]}
\end{tikzcd}$$

Keep these μ_X in mind in the following definition—they serve as the "components" of a natural transformation List $\circ$ List $\to$ List of functors $\mathcal{C} \to \mathcal{D}$, where $\mathcal{C} = \mathcal{D} =$ **Set**.

Definition 4.3.1.2. Let $\mathcal{C}$ and $\mathcal{D}$ be categories and let $F\colon \mathcal{C} \to \mathcal{D}$ and $G\colon \mathcal{C} \to \mathcal{D}$ be functors. A *natural transformation α from F to G*, denoted $\alpha\colon F \to G$, is defined as follows: one announces some constituents (A. components) and asserts that they conform to some laws (1. naturality squares). Specifically, one announces

A. for each object $c \in \mathrm{Ob}(\mathcal{C})$ a morphism $\alpha_c\colon F(c) \to G(c)$ in $\mathcal{D}$, called *the c-component of α.*

One asserts that the following law holds:

1. For every morphism $h\colon c \to c'$ in $\mathcal{C}$, the following square, called the *naturality*

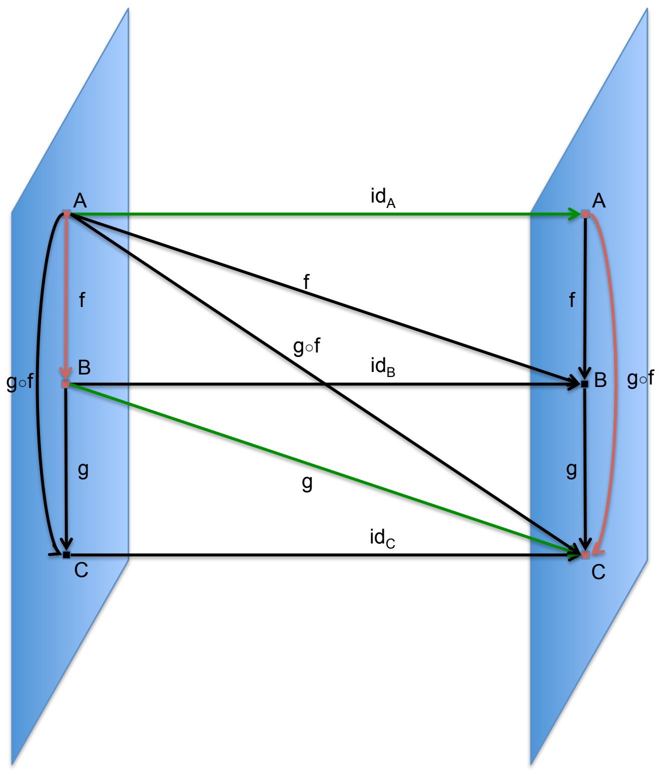
square for h, must commute:

$$\begin{tikzcd} F(c) \arrow[r, "\alpha_c"] \arrow[d, "F(h)"'] \arrow[dr, phantom, "\checkmark"] & G(c) \arrow[d, "G(h)"] \\ F(c') \arrow[r, "\alpha_{c'}"'] & G(c') \end{tikzcd} \tag{4.9}$$

Example 4.3.1.3. Consider the categories $\mathcal{C} \cong [1]$ and $\mathcal{D} \cong [2]$ drawn below:

$$\mathcal{C} := \boxed{\overset{0}{\bullet} \xrightarrow{p} \overset{1}{\bullet}} \qquad \mathcal{D} := \boxed{\overset{A}{\bullet} \xrightarrow{f} \overset{B}{\bullet} \xrightarrow{g} \overset{C}{\bullet}.}$$

Consider the functors $F, G\colon [1] \to [2]$ where $F(0) = A$, $F(1) = B$, $G(0) = A$, and $G(1) = C$. The orange dots and arrows in the picture below represent the image of $\mathcal{C}$ under F and G.



It turns out that there is only one possible natural transformation $F \to G$; we call it α and explore its naturality square. We have drawn the components of $\alpha\colon F \to G$ in green. These components are $\alpha_0 = \mathrm{id}_A\colon F(0) \to G(0)$ and $\alpha_1 = g\colon F(1) \to G(1)$. The naturality square for $p\colon 0 \to 1$ is written twice below, once with notation following that in (4.9) and once in local notation.

$$\begin{tikzcd} F(0) \arrow[r, "\alpha_0"] \arrow[d, "F(p)"'] & G(0) \arrow[d, "G(p)"] \\ F(1) \arrow[r, "\alpha_1"'] & G(1) \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "\mathrm{id}_A"] \arrow[d, "f"'] & A \arrow[d, "g \circ f"] \\ B \arrow[r, "g"'] & C \end{tikzcd}$$

It is clear that this diagram commutes, so our components α_0 and α_1 satisfy the law of Definition 4.3.1.2, making α a natural transformation.

Lemma 4.3.1.4. *Let $\mathcal{C}$ and $\mathcal{D}$ be categories, let $F, G\colon \mathcal{C} \to \mathcal{D}$ be functors, and for every object $c \in \mathrm{Ob}(\mathcal{C})$, let $\alpha_c\colon F(c) \to G(c)$ be a morphism in $\mathcal{D}$. Suppose given a path $c_0 \xrightarrow{f_1} c_1 \xrightarrow{f_2} \cdots \xrightarrow{f_n} c_n$ such that the naturality square*

$$\begin{tikzcd} F(c_{i-1}) \arrow[r, "\alpha_{c_{i-1}}"] \arrow[d, "F(f_i)"'] & G(c_{i-1}) \arrow[d, "G(f_i)"] \\ F(c_i) \arrow[r, "\alpha_{c_i}"'] & G(c_i) \end{tikzcd}$$

commutes for each $1 \leqslant i \leqslant n$. Then the naturality square for the composite $p := f_n \circ \cdots \circ f_2 \circ f_1\colon c_0 \to c_n$

$$\begin{tikzcd} F(c_0) \arrow[r, "\alpha_{c_0}"] \arrow[d, "F(p)"'] & G(c_0) \arrow[d, "G(p)"] \\ F(c_n) \arrow[r, "\alpha_{c_n}"'] & G(c_n) \end{tikzcd}$$

also commutes. In particular, the naturality square commutes for every identity morphism id_c.

Proof. When $n = 0$ we have a path of length 0 starting at each $c \in \mathrm{Ob}(\mathcal{C})$. It vacuously satisfies the condition, so we need to see that its naturality square

$$\begin{tikzcd} F(c) \arrow[r, "\alpha_c"] \arrow[d, "F(\mathrm{id}_c)"'] & G(c) \arrow[d, "G(\mathrm{id}_c)"] \\ F(c) \arrow[r, "\alpha_c"'] & G(c) \end{tikzcd}$$

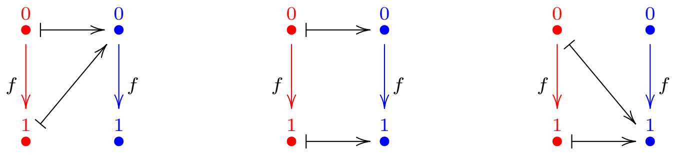
commutes. But this is clear because functors preserve identities.

The rest of the proof follows by induction on n. Suppose $q = f_{n-1} \circ \cdots \circ f_2 \circ f_1\colon c_0 \to c_{n-1}$ and $p = f_n \circ q$ and that the naturality squares for q and for f_n commute; we need only show that the naturality square for p commutes. That is, we assume the two small squares commute below; but it follows that the large rectangle does too, completing the proof.

$$\begin{tikzcd} F(c_0) \arrow[r, "\alpha_{c_0}"] \arrow[d, "F(q)"'] & G(c_0) \arrow[d, "G(q)"] \\ F(c_{n-1}) \arrow[r, "\alpha_{c_{n-1}}"] \arrow[d, "F(f_n)"'] & G(c_{n-1}) \arrow[d, "G(f_n)"] \\ F(c_n) \arrow[r, "\alpha_{c_n}"] & G(c_n) \end{tikzcd}$$

□

Example 4.3.1.5. Let $\mathcal{C} = \mathcal{D} = [1]$ be the linear order of length 1, thought of as a category (by Proposition 4.2.1.17). There are three functors $\mathcal{C} \to \mathcal{D}$, which we can write as $(0,0), (0,1)$, and $(1,1)$; these are depicted left to right below.



These are just functors so far. What are the natural transformations say $\alpha\colon (0,0) \to (0,1)$? To specify a natural transformation, we must specify a component for each object in $\mathcal{C}$. In our case $\alpha_0\colon 0 \to 0$ and $\alpha_1\colon 0 \to 1$. There is only one possible choice: $\alpha_0 = \mathrm{id}_0$ and $\alpha_1 = f$. Now that we have chosen components we need to check the naturality squares.

There are three morphisms in $\mathcal{C}$, namely $\mathrm{id}_0, f, \mathrm{id}_1$. By Lemma 4.3.1.4, we need only check the naturality square for f. We write it twice below, once in the abstract notation and once in concrete notation:

$$\begin{tikzcd}
F(0) \arrow[r, "\alpha_0"] \arrow[d, "F(f)"'] & G(0) \arrow[d, "G(f)"] \\
F(1) \arrow[r, "\alpha_1"'] & G(1)
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
0 \arrow[r, "\mathrm{id}_0"] \arrow[d, "\mathrm{id}_0"'] & 0 \arrow[d, "f"] \\
0 \arrow[r, "f"'] & 1
\end{tikzcd}$$

This commutes, so α is indeed a natural transformation.

Exercise 4.3.1.6. With notation as in Example 4.3.1.5,

a.) how many natural transformations are there $(0,0) \to (1,1)$?

b.) how many natural transformations are there $(0,0) \to (0,0)$?

c.) how many natural transformations are there $(0,1) \to (0,0)$?

d.) how many natural transformations are there $(0,1) \to (1,1)$?

$\lozenge$

Exercise 4.3.1.7. Let $\mathrm{List}\colon \mathbf{Set} \to \mathbf{Set}$ be the functor sending a set X to the set $\mathrm{List}(X)$ of lists with entries in X. We saw above that there is a natural transformation $\mathrm{List} \circ \mathrm{List} \to \mathrm{List}$ given by concatenation.

a.) If someone said “singleton lists give a natural transformation σ from $\mathrm{id}_{\mathbf{Set}}$ to List”, what might they mean? That is, for a set X, what component σ_X might they be suggesting?

b.) Do these components satisfy the necessary naturality squares for functions $f\colon X \to Y$?

$\lozenge$

Exercise 4.3.1.8. Let $\mathcal{C}$ and $\mathcal{D}$ be categories, and suppose that $d \in \mathrm{Ob}(\mathcal{D})$ is a terminal object. Consider the functor $\{d\}^{\mathcal{C}}\colon \mathcal{C} \to \mathcal{D}$ that sends each object $c \in \mathrm{Ob}(\mathcal{C})$ to d and each morphism in $\mathcal{C}$ to the identity morphism id_d on d.

a.) For any other functor $F\colon \mathcal{C} \to \mathcal{D}$, how many natural transformations are there $F \to \{d\}^{\mathcal{C}}$?

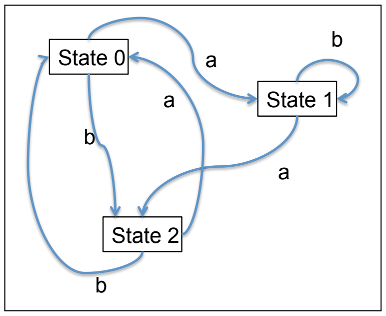
b.) Let $\mathcal{D} = \mathbf{Set}$ and let $d = \{☺\}$. If $\mathcal{C} = [1]$ is the linear order of length 1, and $F\colon \mathcal{C} \to \mathbf{Set}$ is any functor, what does it mean to give a natural transformation $\{d\}^{\mathcal{C}} \to F$?

◊

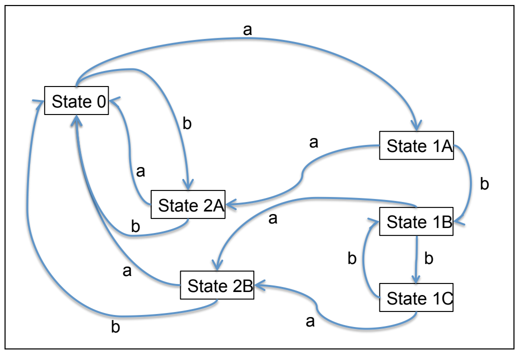
Application 4.3.1.9. In Figure 3.1 we drew a finite state machine on alphabet $\Sigma = \{a, b\}$, and in Example 3.1.3.1 we showed the associated action table. It will be reproduced below. Imagine this was your model for understanding the behavior of some system when acted on by commands a and b. And suppose that a collaborator tells you that she has a more refined notion that fits with the same data. Her notion has 6 states rather than 3, but it's "compatible". What might that mean?

Let's call the original state machine X and the new model Y.

$X :=$



$Y :=$



The action tables for these two machines are:

Original model X		
ID	**a**	**b**
State 0	State 1	State 2
State 1	State 2	State 1
State 2	State 0	State 0

Proposed model Y		
ID	**a**	**b**
State 0	State 1A	State 2A
State 1A	State 2A	State 1B
State 1B	State 2B	State 1C
State 1C	State 2B	State 1B
State 2A	State 0	State 0
State 2B	State 0	State 0

How are these models compatible? Looking at the table for Y, if one removes the distinction between States 1A, 1B, 1C and between States 2A and 2B, then one returns with the table for X. The table for Y is more specific, but it is fully compatible with table X. The sense in which it is compatible is precisely the sense defined by there being a natural transformation.

Recall that $\mathcal{M} = (\text{List}(\Sigma), [\,], +\!\!+)$ is a monoid, and that a monoid is simply a category with one object, say $\text{Ob}(\mathcal{M}) = \{\blacktriangle\}$ (see Section 4.2.1). With $\Sigma = \{a, b\}$, the monoid $\mathcal{M}$ can be visualized as follows:

$$\mathcal{M} = \boxed{\begin{tikzcd} \overset{\blacktriangle}{\bullet} \arrow[loop left, "a"] \arrow[loop right, "b"] \end{tikzcd}}$$

Recall also that a state machine on $\mathcal{M}$ is simply a functor $\mathcal{M} \to$ **Set**. We thus have two such functors, X and Y. A natural transformation $\alpha\colon Y \to X$ would consist of a component α_m for every object $m \in \mathrm{Ob}(\mathcal{M})$, such that certain diagrams commute. But $\mathcal{M}$ having only one object, we need only one function $\alpha_\blacktriangle\colon Y(\blacktriangle) \to X(\blacktriangle)$, where $Y(\blacktriangle)$ is the set of (6) states of Y and $X(\blacktriangle)$ is the set of (3) states of X.

The states of Y have been named so as to make the function $\alpha_\blacktriangle$ particularly easy to guess.[15] We need to check that two squares commute:

$$\begin{tikzcd}
Y(\blacktriangle) \arrow[r, "\alpha_\blacktriangle"] \arrow[d, "Y(a)"'] & X(\blacktriangle) \arrow[d, "X(a)"] \\
Y(\blacktriangle) \arrow[r, "\alpha_\blacktriangle"'] & X(\blacktriangle)
\end{tikzcd}
\qquad
\begin{tikzcd}
Y(\blacktriangle) \arrow[r, "\alpha_\blacktriangle"] \arrow[d, "Y(b)"'] & X(\blacktriangle) \arrow[d, "X(b)"] \\
Y(\blacktriangle) \arrow[r, "\alpha_\blacktriangle"'] & X(\blacktriangle)
\end{tikzcd}
\tag{4.10}$$

This can only be checked by going through and making sure certain things match, as specified by (4.10); we spell it out in gory detail. The columns that should match are those whose entries are written in blue.

Naturality square for $a\colon \blacktriangle \to \blacktriangle$				
$Y(\blacktriangle)$ **[ID]**	$Y(a)$	$\alpha_\blacktriangle \circ Y(a)$	$\alpha_\blacktriangle$	$X(a) \circ \alpha_\blacktriangle$
State 0	State 1A	State 1	State 0	State 1
State 1A	State 2A	State 2	State 1	State 2
State 1B	State 2B	State 2	State 1	State 2
State 1C	State 2B	State 2	State 1	State 2
State 2A	State 0	State 0	State 2	State 0
State 2B	State 0	State 0	State 2	State 0

(4.11)

Naturality square for $b\colon \blacktriangle \to \blacktriangle$				
$Y(\blacktriangle)$ **[ID]**	$Y(b)$	$\alpha_\blacktriangle \circ Y(b)$	$\alpha_\blacktriangle$	$X(b) \circ \alpha_\blacktriangle$
State 0	State 2A	State 2	State 0	State 2
State 1A	State 1B	State 1	State 1	State 1
State 1B	State 1C	State 1	State 1	State 1
State 1C	State 1B	State 1	State 1	State 1
State 2A	State 0	State 0	State 2	State 0
State 2B	State 0	State 0	State 2	State 0

(4.12)

In reality we need to check that for *every* morphism in $\mathcal{M}$, such as $[a, a, b]$, a similar diagram commutes. But this holds automatically. For example (flipping the naturality square sideways for typographical reasons)

$$\begin{tikzcd}
Y(\blacktriangle) \arrow[r, "Y(a)"] \arrow[d, "\alpha_\blacktriangle"] & Y(\blacktriangle) \arrow[r, "Y(a)"] \arrow[d, "\alpha_\blacktriangle"] & Y(\blacktriangle) \arrow[r, "Y(b)"] \arrow[d, "\alpha_\blacktriangle"] & Y(\blacktriangle) \arrow[d, "\alpha_\blacktriangle"] \\
X(\blacktriangle) \arrow[r, "X(a)"'] & X(\blacktriangle) \arrow[r, "X(a)"'] & X(\blacktriangle) \arrow[r, "X(b)"'] & X(\blacktriangle)
\end{tikzcd}$$

[15] The function $\alpha_\blacktriangle\colon Y(\blacktriangle) \to X(\blacktriangle)$ makes the following assignments: State 0 $\mapsto$ State 0, State 1A $\mapsto$ State 1, State 1B $\mapsto$ State 1, State 1C $\mapsto$ State 1, State 2A $\mapsto$ State 2, State 2B $\mapsto$ State 2.

Since each small square above commutes (as checked by tables 4.11 and 4.12), the big outer rectangle commutes too.

To recap, the notion of compatibility between Y and X is one that can be checked and agreed upon by humans, but doing so it is left implicit, and it may be difficult to explain to an outsider what exactly was agreed to, especially in more complex situations. It is quite convenient to simply claim "there is a natural transformation from Y to X."

◊◊

Exercise 4.3.1.10. Let $F\colon \mathcal{C} \to \mathcal{D}$ be a functor. Suppose someone said "the identity on F is a natural transformation from F to itself."

a.) What might they mean?

b.) If it is somehow true, what are the components of this natural transformation?

◊

Example 4.3.1.11. Let $[1] \in \mathrm{Ob}(\mathbf{Cat})$ be the free arrow category described in Exercise 4.1.2.31 and let $\mathcal{D}$ be any category. To specify a functor $F\colon [1] \to \mathcal{D}$ requires the specification of two objects, $F(v_1), F(v_2) \in \mathrm{Ob}(\mathcal{D})$ and a morphism $F(e)\colon F(v_1) \to F(v_2)$ in $\mathcal{D}$. The identity and composition formulas are taken care of once that much is specified. To recap, a functor $F\colon [1] \to \mathcal{D}$ is the same thing as a morphism in $\mathcal{D}$.

Thus, choosing two functors $F, G\colon [1] \to \mathcal{D}$ is precisely the same thing as choosing two morphisms in $\mathcal{D}$. Let us call them $f\colon a_0 \to a_1$ and $g\colon b_0 \to b_1$, where to be clear we have $f = F(e), a_0 = F(v_0), a_1 = F(v_1)$ and $g = G(e), b_0 = G(v_0), b_1 = G(v_1)$.

A natural transformation $\alpha\colon F \to G$ consists of two components, $h_0 := \alpha_{v_0}\colon a_0 \to b_0$ and $h_1 := \alpha_{v_1}\colon a_1 \to b_1$, drawn as dashed lines below:

$$\begin{tikzcd}
a_0 \arrow[r, dashed, "h_0"] \arrow[d, "f"'] & b_0 \arrow[d, "g"] \\
a_1 \arrow[r, dashed, "h_1"'] & b_1
\end{tikzcd}$$

The condition for α to be a natural transformation is that the above square commutes.

In other words, a functor $[1] \to \mathcal{D}$ is an arrow in $\mathcal{D}$ and a natural transformation between two such functors is just a commutative square in $\mathcal{D}$.

Example 4.3.1.12. Recall that to any graph G we can associate the so-called paths-graph $\mathrm{Paths}(G)$, as described in Example 4.1.2.22. This is a functor $\mathrm{Paths}\colon \mathbf{Grph} \to \mathbf{Grph}$. There is also an identity functor $\mathrm{id}_{\mathbf{Grph}}\colon \mathbf{Grph} \to \mathbf{Grph}$. A natural transformation $\eta\colon \mathrm{id}_{\mathbf{Grph}} \to \mathrm{Paths}$ would consist of a graph homomorphism $\eta_G\colon \mathrm{id}_{\mathbf{Grph}}(G) \to \mathrm{Paths}(G)$ for every graph G. But $\mathrm{id}_{\mathbf{Grph}}(G) = G$ by definition, so we need $\eta_G\colon G \to \mathrm{Paths}(G)$. Recall that $\mathrm{Paths}(G)$ has the same vertices as G and every arrow in G counts as a path (of length 1). So there is an obvious graph homomorphism from G to $\mathrm{Paths}(G)$. It is not hard to see that the necessary naturality squares commute.

Example 4.3.1.13. For any graph G we can associate the paths-graph $\mathrm{Paths}(G)$, and nothing stops us from doing that twice to yield a new graph $\mathrm{Paths}(\mathrm{Paths}(G))$. Let's think through what a path of paths in G is. It's a head-to-tail sequence of arrows in $\mathrm{Paths}(G)$, meaning a head-to-tail sequence of paths in G. These composable sequences of paths (or "paths of paths") are the individual arrows in $\mathrm{Paths}(\mathrm{Paths}(G))$. (The vertices in $\mathrm{Paths}(G)$ and $\mathrm{Paths}(\mathrm{Paths}(G))$ are the same as those in G, and all source and target functions are as expected.)

Clearly, given such a sequence of paths in G, we could compose them to one big path in G with the same endpoints. In other words, there is graph morphism $\mu_G\colon \text{Paths}(\text{Paths}(G)) \to \text{Paths}(G)$, that one might call "concatenation". In fact, this concatenation extends to a natural transformation

$$\mu\colon \text{Paths} \circ \text{Paths} \to \text{Paths}$$

between functors **Grph** $\to$ **Grph**. In Example 4.3.1.12, we compared a graph to its paths-graph using a natural transformation $\text{id}_{\mathbf{Grph}} \to \text{Paths}$; here we are making a similar kind of comparison.

Remark 4.3.1.14. In Example 4.3.1.12 we saw that there is a natural transformation sending each graph into its paths-graph. There is a formal sense in which a category is nothing more than a kind of reverse mapping. That is, to specify a category is the same thing as to specify a graph G together with a graph homomorphism $\text{Paths}(G) \to G$. The formalities involve monads, which we will discuss in Section 5.3.

Exercise 4.3.1.15. Let X and Y be sets, and let $f\colon X \to Y$. There is a functor $C_X\colon \mathbf{Grph} \to \mathbf{Set}$ that sends every graph to the set X and sends every morphism of graphs to the identity morphism $\text{id}_X\colon X \to X$. This functor is called *the constant functor at* X. Similarly there is a constant functor $C_Y\colon \mathbf{Grph} \to \mathbf{Set}$.

a.) Use f to construct a natural transformation $C_X \to C_Y$.

b.) What are its components?

◊

Exercise 4.3.1.16. For any graph (V, A, src, tgt) we can extract the set of arrows or the set of vertices. Since each morphism of graphs includes a function between their arrow sets and a function between their vertex sets, we actually have functors $Ar\colon \mathbf{Grph} \to \mathbf{Set}$ and $Ve\colon \mathbf{Grph} \to \mathbf{Set}$.

a.) If someone said "taking source vertices gives a natural transformation from Ar to Ve", what natural transfromation might they be referring to?

b.) What are its components?

c.) If a different person, say from a totally different country, were to say "taking target vertices also gives a natural transformation from Ar to Ve," would they also be correct?

◊

Example 4.3.1.17 (Graph homomorphisms are natural transformations). As discussed above (see Diagram 4.7), there is a category **GrIn** for which a functor $G\colon \mathbf{GrIn} \to \mathbf{Set}$ is the same thing as a graph. Namely, we have

$$\mathbf{GrIn} := \boxed{\begin{tikzcd} \overset{Ar}{\bullet} \arrow[r, shift left, "src"] \arrow[r, shift right, "tgt"'] & \overset{Ve}{\bullet} \end{tikzcd}}$$

A natural transformation of two such functors $\alpha\colon G \to G'$ involves two components, $\alpha_{Ar}\colon G(Ar) \to G'(Ar)$ and $\alpha_{Ve}\colon G(Ve) \to G'(Ve)$, and two naturality squares, one for src and one for tgt. This is precisely the same thing as a graph homomorphism, as defined in Definition 3.3.3.1.

4.3.2 Vertical and horizontal composition

In this section we discuss two types of compositions for natural transformations. The terms vertical and horizontal are used to describe them; these terms come from the following pictures:

$$\begin{tikzcd}
\mathcal{C} \arrow[r, bend left=60, "F"] \arrow[r, "G" description] \arrow[r, bend right=60, "H"'] \arrow[r, phantom, shift left=3ex, "\alpha\Downarrow"] \arrow[r, phantom, shift right=3ex, "\beta\Downarrow"] & \mathcal{D}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{C} \arrow[r, bend left=40, "F_1"] \arrow[r, bend right=40, "G_1"'] \arrow[r, phantom, "\gamma_1\Downarrow"] & \mathcal{D} \arrow[r, bend left=40, "F_2"] \arrow[r, bend right=40, "G_2"'] \arrow[r, phantom, "\gamma_2\Downarrow"] & \mathcal{E}
\end{tikzcd}$$

We generally use $\circ$ to denote both kinds of composition, but if we want to be very clear we will differentiate as follows: $\beta \circ \alpha \colon F \to H$ for vertical composition, and $\gamma_2 \diamond \gamma_1 \colon F_2 \circ F_1 \longrightarrow G_2 \circ G_1$ for horizontal composition. Of course, the actual arrangement of things on a page of text does not correlate with verticality or horizontality—these are just names. We will define them more carefully below.

4.3.2.1 Vertical composition of natural transformations

The following proposition proves that functors and natural transformations (using vertical composition) form a category.

Proposition 4.3.2.2. *Let $\mathcal{C}$ and $\mathcal{D}$ be categories. There exists a category, called* the category of functors from $\mathcal{C}$ to $\mathcal{D}$ *and denoted* $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$*, whose objects are the functors $\mathcal{C} \to \mathcal{D}$ and whose morphisms are the natural transformations,*

$$\mathrm{Hom}_{\mathrm{Fun}(\mathcal{C},\mathcal{D})}(F, G) = \{\alpha \colon F \to G \mid \alpha \text{ is a natural transformation}\}.$$

That is, there are identity natural transformations, natural transformations can be composed, and the identity and associativity laws hold.

Proof. We showed in Exercise 4.3.1.10 that there for any functor $F \colon \mathcal{C} \to \mathcal{D}$, there is an identity natural transformation $\mathrm{id}_F \colon F \to F$ (its component at $c \in \mathrm{Ob}(\mathcal{C})$ is $\mathrm{id}_{F(c)} \colon F(c) \to F(c)$).

Given a natural transformation $\alpha \colon F \to G$ and a natural transformation $\beta \colon G \to H$, we propose for the composite $\beta \circ \alpha$ the transformation $\gamma \colon F \to H$ having components $\beta_c \circ \alpha_c$ for every $c \in \mathrm{Ob}(\mathcal{C})$. To see that γ is indeed a natural transformation, one simply puts together naturality squares for α and β to get naturality squares for $\beta \circ \alpha$.

The associativity and identity laws for $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ follow from those holding for morphisms in $\mathcal{D}$.

□

Notation 4.3.2.3. We sometimes denote the category $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ by $\mathcal{D}^{\mathcal{C}}$.

Example 4.3.2.4. Recall from Exercise 4.1.2.38 that there is a functor $\mathrm{Ob} \colon \mathbf{Cat} \to \mathbf{Set}$ sending a category to its set of objects. And recall from Example 4.1.2.35 that there is a functor $Disc \colon \mathbf{Set} \to \mathbf{Cat}$ sending a set to the discrete category with that set of objects (all morphisms in $Disc(S)$ are identity morphisms). Let $P \colon \mathbf{Cat} \to \mathbf{Cat}$ be the composition $P = Disc \circ \mathrm{Ob}$. Then P takes a category and makes a new category with the same objects but no morphisms. It's like crystal meth for categories.

Let $\mathrm{id}_{\mathbf{Cat}}\colon \mathbf{Cat} \to \mathbf{Cat}$ be the identity functor. There is a natural transformation $i\colon P \to \mathrm{id}_{\mathbf{Cat}}$. For any category $\mathcal{C}$, the component $i_{\mathcal{C}}\colon P(\mathcal{C}) \to \mathcal{C}$ is pretty easily understood. It is a morphism of categories, i.e. a functor. The two categories $P(\mathcal{C})$ and $\mathcal{C}$ have the same set of objects, namely $\mathrm{Ob}(\mathcal{C})$, so our functor is identity on objects; and $P(\mathcal{C})$ has no non-identity morphisms, so nothing else needs be specified.

Exercise 4.3.2.5. Let $\mathcal{C} = \boxed{\overset{A}{\bullet}}$ be the category with $\mathrm{Ob}(\mathcal{C}) = \{A\}$, and $\mathrm{Hom}_{\mathcal{C}}(A, A) = \{\mathrm{id}_A\}$. What is $\mathrm{Fun}(\mathcal{C}, \mathbf{Set})$? In particular, characterize the objects and the morphisms. ◊

Exercise 4.3.2.6. Let $n \in \mathbb{N}$ and let $\underline{n}$ be the set with n elements, considered as a discrete category. [16] In other words, we write $\underline{n}$ to mean what should really be called $Disc(\underline{n})$. Describe the category $\mathrm{Fun}(\underline{3}, \underline{2})$. ◊

Exercise 4.3.2.7. Let $\underline{1}$ denote the discrete category with one object, and let $\mathcal{C}$ be any category.

a.) What are the objects of $\mathrm{Fun}(\underline{1}, \mathcal{C})$?

b.) What are the morphisms of $\mathrm{Fun}(\underline{1}, \mathcal{C})$?

◊

Example 4.3.2.8. Let $\underline{1}$ denote the discrete category with one object (also known as the trivial monoid). For any category $\mathcal{C}$, we investigate the category $\mathcal{D} := \mathrm{Fun}(\mathcal{C}, \underline{1})$. Its objects are functors $\mathcal{C} \to \underline{1}$. Such a functor F assigns to each object in $\mathcal{C}$ an object in $\underline{1}$ of which there is one; so there is no choice in what F does on objects. And there is only one morphism in $\underline{1}$ so there is no choice in what F does on morphisms. The upshot is that there is only one object in $\mathcal{D}$, let's call it F, in $\mathcal{D}$, so $\mathcal{D}$ is a monoid. What are its morphisms?

A morphism $\alpha\colon F \to F$ in $\mathcal{D}$ is a natural transformation of functors. For every $c \in \mathrm{Ob}(\mathcal{C})$ we need a component $\alpha_c\colon F(c) \to F(c)$, which is a morphism $1 \to 1$ in $\underline{1}$. But there is only one morphism in $\underline{1}$, namely id_1, so there is no choice about what these components should be: they are all id_1. The necessary naturality squares commute, so α is indeed a natural transformation. Thus the monoid $\mathcal{D}$ is the trivial monoid; that is, $\mathrm{Fun}(\mathcal{C}, \underline{1}) \cong \underline{1}$ for any category $\mathcal{C}$.

Exercise 4.3.2.9. Let $\underline{0}$ represent the discrete category on 0 objects; it has no objects and no morphisms. Let $\mathcal{C}$ be any category. What is $\mathrm{Fun}(\underline{0}, \mathcal{C})$? ◊

Exercise 4.3.2.10. Let $[1]$ denote the free arrow category as in Exercise 4.1.2.31, and let $\mathcal{C}$ be the graph indexing category from (4.7). Draw the underlying graph of the category $\mathrm{Fun}([1], \mathcal{C})$, and then specify which pairs of paths in that graph correspond to commutative diagrams in $\mathrm{Fun}([1], \mathcal{C})$. ◊

[16] When we have a functor, such as $Disc\colon \mathbf{Set} \to \mathbf{Cat}$, we may sometimes say things like "Let S be a set, considered as a category" (or in general, given a functor $F\colon \mathcal{C} \to \mathcal{D}$, we may say "consider $c \in \mathrm{Ob}(\mathcal{C})$, taken as an object in $\mathcal{D}$"). What this means is that we want to take ideas and methods available in $\mathbf{Cat}$ and use them on our set S. Having our functor $Disc$ lying around, we use it to move S into $\mathbf{Cat}$, as $Disc(S) \in \mathrm{Ob}(\mathbf{Cat})$, upon which we can use our intended methods. However, our human minds get bogged down seeing $Disc(S)$ because it is bulky (e.g. $\mathrm{Fun}(Disc(\underline{3}), Disc(\underline{2}))$ is harder to read than $\mathrm{Fun}(\underline{3}, \underline{2})$). So we abuse notation and write S in place of $Disc(S)$. To add insult to injury, we talk about S as though it was still a set, e.g. discussing its elements rather than its objects. This kind of conceptual abbreviation is standard practice in mathematical discussion because it eases the mental burden for experts, but when one says "Let S be an X considered as a Y" the other may always ask, "How again are you considering X's to be Y's?" and expect a functor .

4.3.2.11 Natural isomorphisms

Let $\mathcal{C}$ and $\mathcal{D}$ be categories. We have defined a category $\text{Fun}(\mathcal{C}, \mathcal{D})$ whose objects are functors $\mathcal{C} \to \mathcal{D}$ and whose morphisms are natural transformations. What are the isomorphisms in this category?

Lemma 4.3.2.12. *Let $\mathcal{C}$ and $\mathcal{D}$ be categories and let $F, G \colon \mathcal{C} \to \mathcal{D}$ be functors. A natural transformation $\alpha \colon F \to G$ is an isomorphism in $\text{Fun}(\mathcal{C}, \mathcal{D})$ if and only if the component $\alpha_c \colon F(c) \to G(c)$ is an isomorphism for each object $c \in \text{Ob}(\mathcal{C})$. In this case α is called a* natural isomorphism*.*

Proof. First suppose that α is an isomorphism with inverse $\beta \colon G \to F$, and let $\beta_c \colon G(c) \to F(c)$ denote its c component. We know that $\alpha \circ \beta = \text{id}_G$ and $\beta \circ \alpha = \text{id}_F$. Using the definitions of composition and identity given in Proposition 4.3.2.2, this means that for every $c \in \text{Ob}(\mathcal{C})$ we have $\alpha_c \circ \beta_c = \text{id}_{G(c)}$ and $\beta_c \circ \alpha_c = \text{id}_{F(c)}$; in other words α_c is an isomorphism.

Second suppose that each α_c is an isomorphism with inverse $\beta_c \colon G(c) \to F(c)$. We need to see that these components assemble into a natural transformation; i.e. for every morphism $h \colon c \to c'$ in $\mathcal{C}$ the right-hand square

$$\begin{tikzcd}
F(c) \arrow[r, "\alpha_c"] \arrow[d, "F(h)"'] \arrow[dr, phantom, "\checkmark"] & G(c) \arrow[d, "G(h)"] \\
F(c') \arrow[r, "\alpha_{c'}"'] & G(c')
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
G(c) \arrow[r, "\beta_c"] \arrow[d, "G(h)"'] \arrow[dr, phantom, "?"] & F(c) \arrow[d, "F(h)"] \\
G(c') \arrow[r, "\beta_{c'}"'] & F(c')
\end{tikzcd}$$

commutes. We know that the left-hand square commutes because α is a natural transformation; we have labeled each square with a ? or a ✓ accordingly. In the following diagram we want to show that the left-hand square commutes. We know that the middle square commutes.

$$\begin{tikzcd}
G(c) \arrow[r, "\beta_c"] \arrow[d, "G(h)"'] \arrow[dr, phantom, "?"] \arrow[rr, bend left, "\text{id}_{G(c)}"] & F(c) \arrow[r, "\alpha_c"] \arrow[d, "F(h)"] \arrow[dr, phantom, "\checkmark"] & G(c) \arrow[r, "\beta_c"] \arrow[d, "G(h)"] \arrow[dr, phantom, "?"] & F(c) \arrow[d, "F(h)"] \\
G(c') \arrow[r, "\beta_{c'}"'] & F(c') \arrow[r, "\alpha_{c'}"'] \arrow[rr, bend right, "\text{id}_{F(c')}"'] & G(c') \arrow[r, "\beta_{c'}"'] & F(c')
\end{tikzcd}$$

To complete the proof we need only to show that $F(h) \circ \beta_c = \beta_{c'} \circ G(h)$. This can be shown by a "diagram chase." We go through it symbolically, for demonstration.

$$F(h) \circ \beta_c = \beta_{c'} \circ \alpha_{c'} \circ F(h) \circ \beta_c = \beta_{c'} \circ G(h) \circ \alpha_c \circ \beta_c = \beta_{c'} \circ G(h).$$

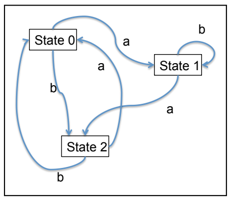
□

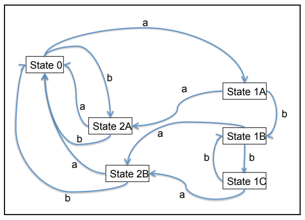
Exercise 4.3.2.13. Recall from Application 4.3.1.9 that a finite state machine on alphabet Σ can be understood as a functor $\mathcal{M} \to \textbf{Set}$, where $\mathcal{M} = \text{List}(\Sigma)$ is the free monoid generated by Σ. In that example we also discussed how natural transformations provide a nice language for changing state machines. Describe what kinds of changes are made by natural isomorphisms. ◊

4.3.2.14 Horizontal composition of natural transformations

Example 4.3.2.15 (Whiskering). Suppose that $\mathcal{M} = \text{List}(a,b)$ and $\mathcal{M}' = \text{List}(m,n,p)$ are free monoids, and let $F\colon \mathcal{M}' \to \mathcal{M}$ be given by sending $[m] \mapsto [a], [n] \mapsto [b]$, and $[p] \mapsto [b,a,a]$. An application of this might be if the sequence $[b,a,a]$ was commonly used in practice and one wanted to add a new button just for that sequence.

Recall Application 4.3.1.9. Let $X\colon \mathcal{M} \to \textbf{Set}$ and $Y\colon \mathcal{M} \to \textbf{Set}$ be the functors, and let $\alpha\colon Y \to X$ be the natural transformation found there. We reproduce them here:





Original model $X\colon \mathcal{M} \to \textbf{Set}$		
ID	**a**	**b**
State 0	State 1	State 2
State 1	State 2	State 1
State 2	State 0	State 0

Proposed model $Y\colon \mathcal{M} \to \textbf{Set}$		
ID	**a**	**b**
State 0	State 1A	State 2A
State 1A	State 2A	State 1B
State 1B	State 2B	State 1C
State 1C	State 2B	State 1B
State 2A	State 0	State 0
State 2B	State 0	State 0

We can compose X and Y with F as in the diagram below

$$\begin{tikzcd}
\mathcal{M}' \arrow[r, "F"] & \mathcal{M} \arrow[r, bend left, "Y"] \arrow[r, bend right, "X"'] \arrow[r, phantom, "\alpha\Downarrow"] & \textbf{Set}
\end{tikzcd}$$

to get functors $Y \circ F$ and $X \circ F$, both of type $\mathcal{M}' \to \textbf{Set}$. What would these be? [17]

$X \circ F$			
ID	**m**	**n**	**p**
State 0	State 1	State 2	State 1
State 1	State 2	State 1	State 0
State 2	State 0	State 0	State 2

$Y \circ F$			
ID	**m**	**n**	**p**
State 0	State 1A	State 2A	State 1A
State 1A	State 2A	State 1B	State 0
State 1B	State 2B	State 1C	State 0
State 1C	State 2B	State 1B	State 0
State 2A	State 0	State 0	State 2A
State 2B	State 0	State 0	State 2A

The map α is what sent both State 1A and State 1B in Y to State 1 in X, and so on. We can see that "the same α works now:" the p column of the table respects that mapping. But α was a natural transformation $Y \to X$ where as we need a natural transformation $Y \circ F \to X \circ F$. This is called *whiskering*. It is a kind of horizontal composition of natural transformation.

[17]The p-column comes from applying b then a then a, as specified above by F.

Definition 4.3.2.16 (Whiskering)**.** Let $\mathcal{B}, \mathcal{C}, \mathcal{D}$, and $\mathcal{E}$ be categories, let $G_1, G_2 \colon \mathcal{C} \to \mathcal{D}$ be functors, and let $\alpha \colon G_1 \to G_2$ a natural transformation. Suppose that $F \colon \mathcal{B} \to \mathcal{C}$ (respectively $H \colon \mathcal{D} \to \mathcal{E}$) is a functor, depicted below:

$$\begin{tikzcd} \mathcal{B} \arrow[r, "F"] & \mathcal{C} \arrow[r, bend left, "G_1"] \arrow[r, bend right, "G_2"'] \arrow[r, phantom, "\alpha\Downarrow"] & \mathcal{D} \end{tikzcd} \qquad \left(\text{respectively,} \qquad \begin{tikzcd} \mathcal{C} \arrow[r, bend left, "G_1"] \arrow[r, bend right, "G_2"'] \arrow[r, phantom, "\alpha\Downarrow"] & \mathcal{D} \arrow[r, "H"] & \mathcal{E} \end{tikzcd} \right),$$

Then the *pre-whiskering of α by F*, denoted $\alpha \diamond F \colon G_1 \circ F \to G_2 \circ F$ (respectively, the *post-whiskering of α by H*, denoted $H \diamond \alpha \colon H \circ G_1 \to H \circ G_2$) is defined as follows.

For each $b \in \mathrm{Ob}(\mathcal{B})$ the component $(\alpha \diamond F)_b \colon G_1 \circ F(b) \to G_2 \circ F(b)$ is defined to be $\alpha_{F(b)}$. (Respectively, for each $c \in \mathrm{Ob}(\mathcal{C})$ the component $(H \diamond \alpha)_c \colon H \circ G_1(c) \to H \circ G_2(c)$ is defined to be $H(\alpha_c)$.) Checking that the naturality squares (in each case) is straightforward.

The rest of this section can safely be skipped; I include it only for my own sense of completeness.

Definition 4.3.2.17 (Horizontal composition of natural transformations)**.** Let $\mathcal{B}, \mathcal{C}$, and $\mathcal{D}$ be categories, let $F_1, F_2 \colon \mathcal{B} \to \mathcal{C}$ and $G_1, G_2 \colon \mathcal{C} \to \mathcal{D}$ be functors, and let $\alpha \colon F_1 \to F_2$ and $\beta \colon G_1 \to G_2$ be natural transformations, as depicted below:

$$\begin{tikzcd} \mathcal{B} \arrow[r, bend left, "F_1"] \arrow[r, bend right, "F_2"'] \arrow[r, phantom, "\alpha\Downarrow"] & \mathcal{C} \arrow[r, bend left, "G_1"] \arrow[r, bend right, "G_2"'] \arrow[r, phantom, "\beta\Downarrow"] & \mathcal{D} \end{tikzcd}$$

By pre- and post-whiskering in one order or the other we get the following diagram

$$\begin{tikzcd} G_1 \circ F_1 \arrow[r, "G_1 \diamond \alpha"] \arrow[d, "\beta \diamond F_1"'] & G_1 \circ F_2 \arrow[d, "\beta \diamond F_2"] \\ G_2 \circ F_1 \arrow[r, "G_2 \diamond \alpha"'] & G_2 \circ F_2 \end{tikzcd}$$

It is straightforward to show that this diagram commutes, so we can take the composition to be our definition of the horizontal composition

$$\beta \diamond \alpha \colon G_1 \circ F_1 \to G_2 \circ F_2.$$

Remark 4.3.2.18. Whiskering a natural transformation α with a functor F is the same thing as horizontally composing α with the identity natural transformation id_F. This is true for both pre- and post- whiskering. For example in the notation of Definition 4.3.2.16 we have

$$\alpha \diamond F = \alpha \diamond \mathrm{id}_F \qquad \text{and} \qquad H \diamond \alpha = \mathrm{id}_H \diamond \alpha.$$

Remark 4.3.2.19. All of the above is somehow similar to the world of paths inside a database schema $\mathcal{S}$, as seen in Definition 3.5.2.3. Indeed, a congruence on the paths of $\mathcal{S}$ is an equivalence relation that is closed under composition. The equivalence relation part is analogous to the fact that natural transformations can be composed vertically. The closure under composition part (Properties (3) and (4) in Definition 3.5.2.3) is analogous to pre- and post whiskering. See also Lemma 3.5.2.5.

This is being mentioned only as a curiosity and a way for the reader to draw connections, not with any additional purpose at this time.

Theorem 4.3.2.20.

$$\begin{tikzcd} \mathcal{C} \arrow[r, bend left=60, "F_1"] \arrow[r, "F_2" description] \arrow[r, bend right=60, "F_3"'] & \mathcal{D} \arrow[r, bend left=60, "G_1"] \arrow[r, "G_2" description] \arrow[r, bend right=60, "G_3"'] & \mathcal{E} \end{tikzcd}$$

with $\alpha_1\Downarrow$, $\alpha_2\Downarrow$, $\beta_1\Downarrow$, $\beta_2\Downarrow$.

Given a setup of categories, functors, and natural transformations as above, we have

$$(\beta_2 \circ \beta_1) \diamond (\alpha_2 \circ \alpha_1) \;=\; (\beta_2 \diamond \alpha_2) \circ (\beta_1 \diamond \alpha_1).$$

Proof. One need only observe that each square in the following diagram commutes, so following the outer path $(\beta_2 \circ \beta_1) \diamond (\alpha_2 \circ \alpha_1)$ yields the same morphism as following the diagonal path ; $(\beta_2 \diamond \alpha_2) \circ (\beta_1 \diamond \alpha_1)$:

$$\begin{tikzcd} G_1F_1 \arrow[r, "G_1\diamond\alpha_1"] \arrow[d, "\beta_1\diamond F_1"'] & G_1F_2 \arrow[r, "G_1\diamond\alpha_2"] \arrow[d, "\beta_1\diamond F_2"'] & G_1F_3 \arrow[d, "\beta_1\diamond F_3"] \\ G_2F_1 \arrow[r, "G_2\diamond\alpha_1"] \arrow[d, "\beta_2\diamond F_1"'] & G_2F_2 \arrow[r, "G_2\diamond\alpha_2"] \arrow[d, "\beta_2\diamond F_2"'] & G_2F_3 \arrow[d, "\beta_2\diamond F_3"] \\ G_3F_1 \arrow[r, "G_3\diamond\alpha_1"'] & G_3F_2 \arrow[r, "G_3\diamond\alpha_2"'] & G_3F_3 \end{tikzcd}$$

□

4.3.3 The category of instances on a database schema

In Section 4.2.2 we showed that schemas are presentations of categories, and we will show in Section 4.4 that in fact the category of schemas is equivalent to the category of categories. In this section we therefore take license to blur the distinction between schemas and categories.

If $\mathcal{C}$ is a schema, i.e. a category, then as we discussed in Section 4.2.2.5, an instance on $\mathcal{C}$ is a functor $I\colon \mathcal{C} \to \mathbf{Set}$. But now we have a notion beyond categories and functors, namely that of natural transformations. So we make the following definition.

Definition 4.3.3.1. Let $\mathcal{C}$ be a schema (or category). The *category of instances on* $\mathcal{C}$, denoted $\mathcal{C}$–$\mathbf{Set}$, is $\mathrm{Fun}(\mathcal{C}, \mathbf{Set})$. Its objects are $\mathcal{C}$-instances (i.e. functors $\mathcal{C} \to \mathbf{Set}$) and its morphisms are natural transformations.

Remark 4.3.3.2. One might object to Definition 4.3.3.1 on the grounds that database instances should not be infinite. This is a reasonable perspective, so it is a pleasant fact that the above definition can be modified easily to accomodate it. The subcategory **Fin** (see Example 4.1.1.4) of finite sets can be substituted for **Set** in Definition 4.3.3.1. One could define the *category of finite instances on* $\mathcal{C}$ as $\mathcal{C} - \mathbf{Fin} = \mathrm{Fun}(\mathcal{C}, \mathbf{Fin})$. Almost all of the ideas in this book will make perfect sense in $\mathcal{C} - \mathbf{Fin}$.

Natural transformations should serve as some kind of morphism between instances on the same schema. How are we to interpret a natural transformation $\alpha\colon I \to J$ between database instances $I, J\colon \mathcal{C} \to \mathbf{Set}$?

Our first clue comes from Application 4.3.1.9. There we considered the case of a monoid $\mathcal{M}$, and we thought about a natural transformation between two functors $X, Y\colon \mathcal{M} \to \mathbf{Set}$, considered as different finite state machines. The notion of natural transformation captured the idea of one model being a refinement of another. This same kind of idea works for databases with more than one table (categories with more than one object), but the whole thing is a bit opaque. Let's work it through slowly.

Example 4.3.3.3. Let us consider the terminal schema, $\underline{1} \cong \boxed{\bullet^{\text{Grapes}}}$. An instance is a functor $\underline{1} \to \mathbf{Set}$ and it is easy to see that this is the same thing as just a set. A natural transformation $\alpha\colon I \to J$ is a function from set I to set J. In the standard table view, we might have I and J as below:

Grapes (I)
ID
Grape 1
Grape 3
Grape 4

Grapes (J)
ID
Jan1-01
Jan1-02
Jan1-03
Jan1-04
Jan3-01
Jan4-01
Jan4-02

There are 343 natural transformations $I \to J$. Perhaps some of them make more sense than others; e.g. we could hope that the numbers in I corresponded to the numbers after the dash in J, or perhaps to what seems to be the date in January. But it could be that the rows in J correspond to batches, and all three grapes in I are part of the first batch on Jan-1. The notion of natural transformation is a mathematical one.

Exercise 4.3.3.4. Recall the notion of set-indexed sets from Definition 2.7.6.12. Let A be a set, and come up with a schema $\mathcal{A}$ such that instances on $\mathcal{A}$ are A-indexed sets. Is our current notion of morphism between instances (i.e. natural transformations) well-aligned with the above definition of "mapping of A-indexed sets"? ◊

For a general schema (or category) $\mathcal{C}$, let us think through what a morphism $\alpha\colon I \to J$ between instances $I, J\colon \mathcal{C} \to \mathbf{Set}$ is. For each object $c \in \mathrm{Ob}(\mathcal{C})$ there is a component $\alpha_c\colon I(c) \to J(c)$. This means that just like in Example 4.3.3.3, there is for each table c a function from the rows in I's manifestation of c to the rows in J's manifestation of c. So to make a natural transformation, such a function has to be specified table by table. But then we have to contend with naturality squares, one for every arrow in $\mathcal{C}$. Arrows in $\mathcal{C}$ correspond to foreign key columns in the database. The naturality requirement was already covered in Application 4.3.1.9 (and see especially how (4.10) is checked in (4.11) and (4.12)).

Example 4.3.3.5. We saw in Section 4.2.1.20 that graphs can be regarded as functors $\mathcal{G} \to \mathbf{Set}$, where $\mathcal{G} \cong \mathbf{GrIn}$ is the "schema for graphs" shown here:

$$\mathcal{G} := \boxed{\begin{tikzcd} \overset{\texttt{Arrow}}{\bullet} \arrow[r, shift left, "src"] \arrow[r, shift right, "tgt"'] & \overset{\texttt{Vertex}}{\bullet} \end{tikzcd}}$$

A database instance $I\colon \mathcal{G} \to \mathbf{Set}$ on $\mathcal{G}$ consists of two tables. Here is an example instance:

$$I := \begin{tikzcd} \bullet^{v} \arrow[r, "f"] & \bullet^{w} \arrow[r, bend left, "g"] \arrow[r, bend right, "h"'] & \bullet^{x} \end{tikzcd}$$

`Arrow` (I)		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x

`Vertex` (I)
ID
v
w
x

To discuss natural transformations, we need two instances. Here is another, $J\colon \mathcal{G} \to \mathbf{Set}$,

$$J := \begin{tikzcd} \overset{q}{\bullet} \arrow[r, "i"] & \overset{r}{\bullet} \arrow[r, bend left, "j"] & \overset{s}{\bullet} \arrow[l, bend left, "k"] \arrow[r, "\ell"] & \overset{t}{\bullet} \\ & & & \overset{u}{\bullet} \end{tikzcd}$$

`Arrow` (J)		
ID	**src**	**tgt**
i	q	r
j	r	s
k	s	r
ℓ	s	t

`Vertex` (J)
ID
q
r
s
t
u

To give a natural transformation $\alpha\colon I \to J$, we give two components: one for arrows and one for vertices. We need to say where each vertex in I goes in J and we need to say where each arrow in I goes in J. The naturality squares insist that if we specify that $g \mapsto j$, for example, then we better specify that$w \mapsto r$ and that $x \mapsto s$. What a computer is very good at, but a human is fairly slow at, is checking that a given pair of components (arrows and vertices) really is natural.

There are 8000 ways to come up with component functions $\alpha_{\texttt{Arrow}}$ and $\alpha_{\texttt{Vertex}}$, but precisely four natural transformations, i.e. four graph homomorphisms, $I \to J$; the other 7996 are haphazard flingings of arrows to arrows and vertices to vertices without any regard to sources and targets. We briefly describe the four now.

First off, nothing can be sent to u because arrows must go to arrows and u touches no arrows. If we send $v \mapsto q$ then f must map to i, and w must map to r, and both g and h must map to j, and x must map to s. If we send $v \mapsto r$ then there are two choices for g and h. If we send $v \mapsto s$ then there's one way to obtain a graph morphism. If we try to send $v \overset{?}{\mapsto} t$, we fail. All of this can be seen by staring at the tables rather than at the pictorial representations of the graphs; the human eye understands these pictures better, but the computer understands the tables better.

Exercise 4.3.3.6. If $I, J\colon \mathcal{G} \to \mathbf{Set}$ are as in Example 4.3.3.5, how many natural transformations are there $J \to I$? ◊

Exercise 4.3.3.7. Let $Y_A\colon \mathcal{G} \to \mathbf{Set}$ denote the instance below:

`Arrow` (Y_A)		
ID	**src**	**tgt**
a	v_0	v_1

`Vertex` (Y_A)
ID
v_0
v_1

Let $I\colon \mathcal{G} \to \mathbf{Set}$ be as in Example 4.3.3.5.

a.) How many natural transformations are there $Y_A \to I$?

b.) With J as above, how many natural transformations are there $Y_A \to J$?

c.) Do you have any conjecture about the way natural transformations $Y_A \to X$ behave for arbitrary graphs $X\colon \mathcal{G} \to \mathbf{Set}$?

◊

In terms of databases, this notion of instance morphism $I \to J$ is fairly benign. For every table its a mapping from the set of rows in I's version of the table to J's version of the table, such that all the foreign keys are respected. We will see that this notion of morphism has excellent formal properties, so that projections, unions, and joins of tables (the typical database operations) would be predicted to be "obviously interesting" by a category theorist who had no idea what a database was. [18]

However, something is also missing from the natural transformation picture. A very important occurrence in the world of databases is the update. Everyone can understand this: a person makes a change in one of the tables, like changing your address from Cambridge, MA to Hereford, UK. Most such arbitrary changes of database instance are not "natural", in that the new linking pattern is incompatible with the old.

It is interesting to consider how updates of $\mathcal{C}$-instances should be understood category theoretically. We might want a category $Upd_{\mathcal{C}}$ whose objects are $\mathcal{C}$-instances and whose morphisms are updates. But then what is the composition formula? Is there a unique morphism $I \to J$ whenever J can be obtained as an update on I? Because in that case, we would be defining $Upd_{\mathcal{C}}$ to be the indiscrete category on the set of $\mathcal{C}$-instances (see Example 4.3.4.3).

Exercise 4.3.3.8. Research project: Can you come up with a satisfactory way to model database updates category-theoretically? Let $\mathbb{N}$ be the category

$$\begin{tikzcd}[\mathbb{N}] := & \overset{0}{\bullet} \arrow[r] & \overset{1}{\bullet} \arrow[r] & \overset{2}{\bullet} \arrow[r] & \cdots\end{tikzcd}$$

representing a discrete timeline. A place to start might be to use something like the slice category $\mathbf{Cat}_{/[\mathbb{N}]}$ where the fiber over each object in $\mathbb{N}$ is a snapshot of the database in time. Can you make this work? ◊

4.3.4 Equivalence of categories

We have a category **Cat** of categories, and in every category there is a notion of isomorphism between objects: one morphism each way, such that each round-trip composition is the identity. An isomorphism in **Cat**, therefore, takes place between two categories, say $\mathcal{C}$ and $\mathcal{D}$: it is a functor $F\colon \mathcal{C} \to \mathcal{D}$ and a functor $G\colon \mathcal{D} \to \mathcal{C}$ such that $G \circ F = \mathrm{id}_{\mathcal{C}}$ and $F \circ G = \mathrm{id}_{\mathcal{D}}$.

It turns out that categories are often similar enough to be considered equivalent without being isomorphic. For this reason, the notion of isomorphism is considered "too strong" to be useful for categories. The feeling to a category theorist might be akin to saying that two material samples are the same if there is an atom-by-atom matching, or that two words are the same if they are written in the same font, of the same size, by the same person, in the same state of mind.

As reasonable as isomorphism is as a notion *in* most categories, it fails to be the "right notion" *about* categories. The reason is that *in* categories there are objects and morphisms, whereas when we talk *about* categories, we have categories and functors, plus natural transformations. These serve as mappings between mappings, and this is not part of the structure of an ordinary category. In cases where a category $\mathcal{C}$ does have such mappings between mappings, it is often a "better notion" if we take that extra

[18] More precisely, given a functor between schemas $F\colon \mathcal{C} \to \mathcal{D}$, the pullback $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$, its left Σ_F and its right adjoint Π_F constitute these important queries. See Section 5.1.4.

structure into account, like we will for categories. This whole subject leads us to the study of 2-categories (or n-categories, or ∞-categories), which we do not discuss in this book. See, for example, [Le1] for an introduction.

Regardless, our purpose now is to explain this "good notion" of sameness for categories, namely *equivalences of categories*, which appropriately take natural transformations into account. Instead of "functors going both ways with round trips equal to identity", which is required in order to be an isomorphism of categories, equivalence of categories demands "functors going both ways with round trips *isomorphic* to identity".

Definition 4.3.4.1 (Equivalence of categories)**.** Let $\mathcal{C}$ and $\mathcal{C}'$ be categories. A functor $F\colon \mathcal{C} \to \mathcal{C}'$ is called *an equivalence of categories*, and denoted $F\colon \mathcal{C} \xrightarrow{\simeq} \mathcal{C}'$, [19] if there exists a functor $F'\colon \mathcal{C}' \to \mathcal{C}$ and natural isomorphisms $\alpha\colon \mathrm{id}_{\mathcal{C}} \xrightarrow{\cong} F' \circ F$ and $\alpha'\colon \mathrm{id}_{\mathcal{C}'} \xrightarrow{\cong} F \circ F'$. In this case we say that F and F' are *mutually inverse equivalences.*

Unpacking a bit, suppose we are given functors $F\colon \mathcal{C} \to \mathcal{C}'$ and $F'\colon \mathcal{C}' \to \mathcal{C}$. We want to know something about the roundtrips on $\mathcal{C}$ and on $\mathcal{C}'$; we want to know the same kind of information about each roundtrip, so let's concentrate on the $\mathcal{C}$ side. We want to know something about $F' \circ F\colon \mathcal{C} \to \mathcal{C}$, so let's name it $i\colon \mathcal{C} \to \mathcal{C}$; we want to know that i is a natural isomorphism. That is, for every $c \in \mathrm{Ob}(\mathcal{C})$ we want an isomorphism $\alpha_c\colon c \xrightarrow{\cong} i(c)$, and we want to know that these isomorphisms are picked carefully enough that given $g\colon c \to c'$ in $\mathcal{C}$, the choice of isomorphisms for c and c' are compatible,

$$\begin{tikzcd}
c \arrow[r, "\alpha_c"] \arrow[d, "g"'] & i(c) \arrow[d, "i(g)"] \\
c' \arrow[r, "\alpha_{c'}"'] & i(c').
\end{tikzcd}$$

To be an equivalence, the same has to hold for the other roundtrip, $i' = F \circ F'\colon \mathcal{C}' \to \mathcal{C}'$.

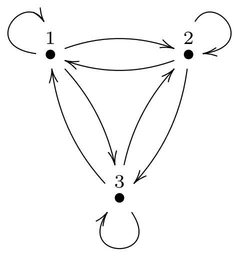
Exercise 4.3.4.2. Let $\mathcal{C}$ and $\mathcal{C}'$ be categories. Suppose that $F\colon \mathcal{C} \to \mathcal{C}'$ is an isomorphism of categories.

a.) Is it an equivalence of categories?

b.) What are the components of α and α' (with notation as in Definition 4.3.4.1)?

◊

Example 4.3.4.3. Let S be a set and let $S \times S \subseteq S \times S$ be the complete relation on S, which is a preorder K_S. Recall from Proposition 4.2.1.17 that we have a functor $i\colon \mathbf{PrO} \to \mathbf{Cat}$, and the resulting category $i(K_S)$ is called the *indiscrete category on S*; it has objects S and a single morphism between every pair of objects. Here is a picture of $K_{\{1,2,3\}}$:



[19]The notation $\simeq$ has already been used for equivalences of paths in a schema. We do not mean to equate these ideas; we are just reusing the symbol. Hopefully no confusion will arise.

It is easy check that $K_{\underline{1}}$, the indiscrete category on one element, is isomorphic to $\underline{1}$, the discrete category on one object, also known as the terminal category (see Exercise 4.1.2.37). The category $\underline{1}$ consists of one object, its identity morphism, and nothing else.

The only way that K_S can be isomorphic to $\underline{1}$ is if S has one element. [20] On the other hand, there is an equivalence of categories

$$K_S \simeq \underline{1}$$

for every set $S \neq \varnothing$.

In fact, there are many such equivalences, one for each element of S. To see this, let S be a nonempty set and choose an element $s_0 \in S$. For every $s \in S$, there is a unique isomorphism $k_s \colon s \xrightarrow{\cong} s_0$ in K_S. Let $F \colon K_S \to \underline{1}$ be the only possible functor (see Exercise 4.1.2.37), and let $F' \colon \underline{1} \to K_S$ send the unique object in $\underline{1}$ to the object s_0.

Note that $F' \circ F = \mathrm{id}_{\underline{1}} \colon \underline{1} \to \underline{1}$ is the identity, but that $F \circ F' \colon K_S \to K_S$ sends everything to s_0. Let $\alpha = \mathrm{id}_{\underline{1}}$ and define $\alpha' \colon \mathrm{id}_{K_S} \to F \circ F'$ by $\alpha'_s = k_s$. Note that α'_s is an isomorphism for each $s \in \mathrm{Ob}(K_S)$, and note that α' is a natural transformation (hence natural isomorphism) because every possible square commutes in K_S. This completes the proof, initiated in the paragraph above, that the category K_S is equivalent to $\underline{1}$ for every nonempty set S, and that this fact can be witnessed by any element $s_0 \in S$.

Example 4.3.4.4. Consider the category **FLin**, described in Example 4.1.1.11, of finite nonempty linear orders. For every natural number $n \in \mathbb{N}$, let $[n] \in \mathrm{Ob}(\mathbf{FLin})$ denote the linear order shown in Example 3.4.1.7. Define a category $\mathbf{\Delta}$ whose objects are given by $\mathrm{Ob}(\mathbf{\Delta}) = \{[n] \mid n \in \mathbb{N}\}$ and with $\mathrm{Hom}_{\mathbf{\Delta}}([m],[n]) = \mathrm{Hom}_{\mathbf{FLin}}([m],[n])$. The difference between **FLin** and $\mathbf{\Delta}$ is only that objects in **FLin** may have "funny labels", e.g.

$$\begin{tikzcd} \overset{5}{\bullet} \arrow[r] & \overset{x}{\bullet} \arrow[r] & \overset{\text{“}Sam\text{”}}{\bullet} \end{tikzcd}$$

whereas objects in $\mathbf{\Delta}$ all have standard labels, e.g.

$$\begin{tikzcd} \overset{0}{\bullet} \arrow[r] & \overset{1}{\bullet} \arrow[r] & \overset{2}{\bullet} \end{tikzcd}$$

Clearly **FLin** is a much larger category, and yet feels like it is "pretty much the same as" $\mathbf{\Delta}$. Justly, they are equivalent, $\mathbf{FLin} \simeq \mathbf{\Delta}$.

The functor $F' \colon \mathbf{\Delta} \to \mathbf{FLin}$ is the inclusion; the functor $F \colon \mathbf{FLin} \to \mathbf{\Delta}$ sends every finite nonempty linear order $X \in \mathrm{Ob}(\mathbf{FLin})$ to the object $F(X) := [n] \in \mathbf{\Delta}$, where $\mathrm{Ob}(X) \cong \{0, 1, \ldots, n\}$. For each such X there is a unique isomorphism $\alpha_X \colon X \xrightarrow{\cong} [n]$, and these fit together into [21] the required natural isomorphism $\mathrm{id}_{\mathbf{FLin}} \to F' \circ F$. The other natural isomorphism $\alpha' \colon \mathrm{id}_{\mathbf{\Delta}} \to F \circ F'$ is the identity.

Exercise 4.3.4.5. Recall from Definition 2.1.2.16 that a set X is called finite if there exists a natural number $n \in \mathbb{N}$ and an isomorphism of sets $X \to \underline{n}$. Let **Fin** denote the category whose objects are the finite sets and whose morphisms are the functions. Let $\mathcal{S}$ denote the category whose objects are the sets $\underline{n}$ and whose morphisms are again the functions. For every object $X \in \mathrm{Ob}(\mathbf{Fin})$ there exists an isomorphism $p_X \colon X \to \underline{n}$ for some unique object $\underline{n} \in \mathrm{Ob}(\mathcal{S})$. Find an equivalence of categories $\mathbf{Fin} \xrightarrow{\simeq} \mathcal{S}$. ◊

[20] One way to see this is that by Exercise 4.1.2.38, we have a functor $\mathrm{Ob} \colon \mathbf{Cat} \to \mathbf{Set}$, and we know by Exercise 4.1.2.24 that functors preserve isomorphisms, so an isomorphism between categories must restrict to an isomorphism between their sets of objects. The only sets that are isomorphic to $\underline{1}$ have one element.

[21] The phrase "these fit together into" is suggestive shorthand for, and thus can be replaced with, the phrase "the naturality squares commute for these components, so together they constitute".

Exercise 4.3.4.6. We say that two categories $\mathcal{C}$ and $\mathcal{D}$ are equivalent if there exists an equivalence of categories between them. Show that the relation of "being equivalent" is an equivalence relation on $\mathrm{Ob}(\mathbf{Cat})$. ◊

Example 4.3.4.7. Consider the group $\mathbb{Z}_2 := (\{0,1\}, 0, +)$, where $1+1=0$. As a category, $\mathbb{Z}_2$ has one object ▲ and two morphisms, namely $0, 1$, such that 0 is the identity. Since $\mathbb{Z}_2$ is a group, the morphism $1\colon ▲ \to ▲$ must have an inverse x, meaning $1+x=0$, and $x=1$ is the only solution.

The point is that the morphism 1 in $\mathbb{Z}_2$ is an isomorphism. Let $\mathcal{C} = \underline{1}$ be the terminal category as in Exercise 4.1.2.37. One might accidentally believe that $\mathcal{C}$ is equivalent to $\mathbb{Z}_2$, but this is not the case! The argument in favor of the accidental belief is that we have unique functors $F\colon \mathbb{Z}_2 \to \mathcal{C}$ and $F'\colon \mathcal{C} \to \mathbb{Z}_2$ (and this is true); the roundtrip $F \circ F'\colon \mathcal{C} \to \mathcal{C}$ is the identity (and this is true); and for the roundtrip $F' \circ F\colon \mathbb{Z}_2 \to \mathbb{Z}_2$ both morphisms in $\mathbb{Z}_2$ are isomorphisms, so any choice of morphism $\alpha_▲\colon ▲ \to F' \circ F(▲)$ will be an isomorphism (and this is true). The problem is that no such $\alpha_▲$ will be a natural transformation.

When we roundtrip $F' \circ F\colon \mathbb{Z}_2 \to \mathbb{Z}_2$, the image of $1\colon ▲ \to ▲$ is $F' \circ F(1) = 0 = \mathrm{id}_▲$. So the naturality square for the morphism 1 looks like this:

$$\begin{tikzcd} \blacktriangle \arrow[r, "\alpha_\blacktriangle"] \arrow[d, "1"'] & \blacktriangle \arrow[d, "0=F'\circ F(1)"] \\ \blacktriangle \arrow[r, "\alpha_\blacktriangle"'] & \blacktriangle \end{tikzcd}$$

where we still haven't decided whether we want $\alpha_▲$ to be 0 or 1. Unfortunately, neither choice works (i.e. for neither choice will the diagram commute) because $x+1 \neq x+0$ in $\mathbb{Z}_2$.

Definition 4.3.4.8 (Skeleton)**.** Let $\mathcal{C}$ be a category. We saw in Lemma 4.1.1.21 that the relation of "being isomorphic" is an equivalence relation $\cong$ on $\mathrm{Ob}(\mathcal{C})$. An *election in* $\mathcal{C}$ is a choice E of the following sort:

- for each $\cong$-equivalence class $S \subseteq \mathrm{Ob}(\mathcal{C})$ a choice of object $s_E \in S$, called the *elected object for* S, and
- for each object $c \in \mathrm{Ob}(\mathcal{C})$ a choice of isomorphism $i_c\colon s_E \to c$ and $j_c\colon c \to s_E$ with $i_c \circ j_c = \mathrm{id}_c$ and $j_c \circ i_c = \mathrm{id}_{s_E}$, where s_E is an elected object (depending on c).

Given an election E in $\mathcal{C}$, there is a category called the *E-elected skeleton of* $\mathcal{C}$, denoted $\mathrm{Skel}_E(\mathcal{C})$, whose objects are the elected objects and whose morphisms $s \to t$ for any elected objects $s, t \in \mathrm{Ob}(\mathcal{C})$ are given by $\mathrm{Hom}_{\mathrm{Skel}_E(\mathcal{C})}(s,t) = \mathrm{Hom}_{\mathcal{C}}(s,t)$. Any object $c \in \mathrm{Ob}(\mathcal{C})$ is isomorphic to a unique elected object s_E; we refer to s_E as the *elected representative* of c; we refer to the isomorphisms i_c and j_c as the *representing isomorphisms* for c.

Proposition 4.3.4.9. *Let $\mathcal{C}$ be a category and let E be an election in $\mathcal{C}$. There is an equivalence of categories*

$$\mathrm{Skel}_E(\mathcal{C}) \simeq \mathcal{C}.$$

Proof. The functor $F'\colon \mathrm{Skel}_E(\mathcal{C}) \to \mathcal{C}$ is the inclusion. The functor $F\colon \mathcal{C} \to \mathrm{Skel}_E(\mathcal{C})$ sends each object in $\mathcal{C}$ to its elected representative. Given objects $c, c' \in \mathrm{Ob}(\mathcal{C})$ with elected representatives s, t respectively, and given a morphism $g\colon c \to c'$ in $\mathcal{C}$, let $i_c, j_c, i_{c'}$, and $j_{c'}$ be the representing isomorphisms, and define $F(g)\colon s \to t$ to be the composite

$$\begin{tikzcd} s \arrow[r, "i_c"] & c \arrow[r, "g"] & c' \arrow[r, "j_{c'}"] & t. \end{tikzcd}$$

This is functorial because it sends the identity to the identity and $F(g \circ g') = F(g) \circ F(g')$.

The composite $F \circ F'\colon \text{Skel}_E(\mathcal{C}) \to \text{Skel}_E(\mathcal{C})$ is the identity. For each $c \in \text{Ob}(\mathcal{C})$ define $\alpha_c\colon c \xrightarrow{\cong} F' \circ F(c)$ by $\alpha_c := j_c$. Given $g\colon c \to c'$ the required naturality square is shown to the left below:

$$\begin{tikzcd} c \arrow[r, "j_c"] \arrow[d, "g"'] \arrow[dr, phantom, "?"] & s \arrow[r, "i_c"] \arrow[d, "F' \circ F(g)"] & c \arrow[d, "g"] \\ c' \arrow[r, "j_c'"'] & t & c' \arrow[l, "j_c'"] \end{tikzcd}$$

The right-hand part commutes by definition of F and F'; i.e. $j' \circ g \circ i_c = F' \circ F(g)$. The left-hand square commutes because $i_c \circ j_c = \text{id}_c$.

□

Definition 4.3.4.10. A *skeleton of* $\mathcal{C}$ is a category $\mathcal{S}$, equivalent to $\mathcal{C}$, such that for any two objects $s, s' \in \text{Ob}(\mathcal{S})$, if $s \cong s'$ then $s = s'$.

Exercise 4.3.4.11. Let $\mathcal{P}$ be a preorder (considered as a category).

a.) If $\mathcal{P}'$ is a skeleton of $\mathcal{P}$, is it a partial order?

b.) Is every partial order the skeleton of some preorder?

◊

Definition 4.3.4.12 (Full and faithful functors)**.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories, and let $F\colon \mathcal{C} \to \mathcal{D}$ be a functor. For any two objects $c, c' \in \text{Ob}(\mathcal{C})$, we have a function $\text{Hom}_F(c, c')\colon \text{Hom}_{\mathcal{C}}(c, c') \to \text{Hom}_{\mathcal{D}}(F(c), F(c'))$ guaranteed by the definition of functor. We say that F is *a full functor* if $\text{Hom}_F(c, c')$ is surjective for every c, c'. We say that F is *a faithful functor* if $\text{Hom}_F(c, c')$ is injective for every c, c'. We say that F is *a fully faithful functor* if $\text{Hom}_F(c, c')$ is bijective for every c, c'.

Exercise 4.3.4.13. Let $\underline{1}$ and $\underline{2}$ be the discrete categories on one and two objects, respectively. There is only one functor $\underline{2} \to \underline{1}$.

a.) Is it full?

b.) Is it faithful?

◊

Exercise 4.3.4.14. Let $\underline{0}$ denote the empty category, and let $\mathcal{C}$ be any category. There is a unique functor $F\colon \underline{0} \to \mathcal{C}$.

a.) For general $\mathcal{C}$ will F be full?

b.) For general $\mathcal{C}$ will F be faithful?

c.) For general $\mathcal{C}$ will F be an equivalence of categories?

◊

Proposition 4.3.4.15. *Let $\mathcal{C}$ and $\mathcal{C}'$ be categories and let $F\colon \mathcal{C} \to \mathcal{C}'$ be an equivalence of categories. Then F is fully faithful.*

Proof. Suppose F is an equivalence, so we can find a functor $F'\colon \mathcal{C}' \to \mathcal{C}$ and natural isomorphisms $\alpha\colon \mathrm{id}_{\mathcal{C}} \xrightarrow{\cong} F' \circ F$ and $\alpha'\colon \mathrm{id}_{\mathcal{C}'} \xrightarrow{\cong} F \circ F'$. We need to know that for any objects $c, d \in \mathrm{Ob}(\mathcal{C})$, the map

$$\mathrm{Hom}_F(c,d)\colon \mathrm{Hom}_{\mathcal{C}}(c,d) \to \mathrm{Hom}_{\mathcal{C}'}(Fc, Fd)$$

is bijective. Consider the following diagram

$$\begin{tikzcd}
\mathrm{Hom}_{\mathcal{C}}(c,d) \arrow[r, "\mathrm{Hom}_F(c,d)"] \arrow[dr, "\alpha"'] & \mathrm{Hom}_{\mathcal{C}'}(Fc,Fd) \arrow[d, "\mathrm{Hom}_{F'}(Fc,Fd)"] \arrow[dr, "\alpha'"] & \\
& \mathrm{Hom}_{\mathcal{C}}(F'Fc, F'Fd) \arrow[r, "\mathrm{Hom}_F(F'Fc,F'Fd)"'] & \mathrm{Hom}_{\mathcal{C}'}(FF'Fc, FF'Fd)
\end{tikzcd}$$

The fact that α is bijective implies that the vertical function is surjective. The fact that α' is bijective implies that the vertical function is injective, so it is bijective. This implies that $\mathrm{Hom}_F(c,d)$ is bijective as well.

□

Exercise 4.3.4.16. Let $\mathbb{Z}_2$ be the group (as category) from Example 4.3.4.7. Are there any fully faithful functors $\mathbb{Z}_2 \to \underline{1}$? ◊

4.4 Categories and schemas are equivalent, Cat $\simeq$ Sch

Perhaps it is intuitively clear that schemas are somehow equivalent to categories, and in this section we make that precise. The basic idea was already laid out in Section 4.2.2.

4.4.1 The category Sch of schemas

Recall from Definition 3.5.2.6 that a schema consists of a pair $\mathcal{C} := (G, \simeq)$, where $G = (V, A, src, tgt)$ is a graph and $\simeq$ is a congruence, meaning a kind of equivalence relation on the paths in G (see Definition 3.5.2.3. If we think of a schema as being analogous to a category, what should fulfill the role of functors? That is, what are to be the morphisms in **Sch**?

Unfortunately, ones first guess may give the wrong notion if we want an equivalence **Sch** $\simeq$ **Cat**. Since objects in **Sch** are graphs with additional structure, one might imagine that a morphism $\mathcal{C} \to \mathcal{C}'$ in **Sch** should be a graph homomorphism (as in Definition 3.3.3.1) that preserves said structure. But graph homomorphisms require that arrows be sent to arrows, whereas we are more interested in paths than in individual arrows—the arrows are merely useful for presentation.

If instead we define morphisms between schemas to be maps that send paths in $\mathcal{C}$ to paths in $\mathcal{C}'$, subject to the requirements that path endpoints, path concatenations, and path equivalences are preserved, this will turn out to give the correct notion. And since a path is a concatenation of its arrows, it suffices to give a function F from the arrows of $\mathcal{C}$ to the paths of $\mathcal{C}'$, which automatically takes care of the first two requirements above; we must only take care that F preserves path equivalences.

Recall from Examples 4.1.2.22 and 4.3.1.13 the paths-graph functor $\text{Paths}\colon \mathbf{Grph} \to \mathbf{Grph}$, the paths of paths functor $\text{Paths} \circ \text{Paths}\colon \mathbf{Grph} \to \mathbf{Grph}$, and the natural transformations for any graph G,

$$\eta_G\colon G \to \text{Paths}(G) \qquad \text{and} \qquad \mu_G\colon \text{Paths}(\text{Paths}(G)) \to \text{Paths}(G). \tag{4.13}$$

The function η_G spells out the fact that every arrow in G counts as a path in G, and the function μ_G spells out the fact that a head-to-tail sequence of paths (a path of paths) in G can be concatenated to a single path in G.

Exercise 4.4.1.1. Let $[2]$ denote the graph $\overset{0}{\bullet} \to \overset{1}{\bullet} \to \overset{2}{\bullet}$, and let $\mathcal{L}oop$ denote the unique graph having one vertex and one arrow (pictured in Diagram (3.17)).

a.) Find a graph homomorphism $f\colon [2] \to \text{Paths}(\mathcal{L}oop)$ that is injective on arrows (i.e. such that no two arrows in the graph $[2]$ are sent by f to the same arrow in $\text{Paths}(\mathcal{L}oop)$).

b.) The graph $[2]$ has 6 paths, so $\text{Paths}([2])$ has 6 arrows. What are the images of these arrows under the graph homomorphism $\text{Paths}(f)\colon \text{Paths}([2]) \to \text{Paths}(\text{Paths}(\mathcal{L}oop))$?

◊

We are almost ready to give the definition of schema morphism, but before we do, let's return to our original idea. Given graphs G, G' (underlying schemas $\mathcal{C}, \mathcal{C}'$) we originally wanted a function from the paths in G to the paths in G', but we realized it was more concise to speak of a function from arrows in G to paths in G'. How do we get back what we originally wanted from the concise version? Given a graph homomorphism $f\colon G \to \text{Paths}(G')$, we use (4.13) to form the following composition, which we denote simply by $\text{Paths}_f\colon \text{Paths}(G) \to \text{Paths}(G')$:

$$\text{Paths}(G) \xrightarrow{\text{Paths}(f)} \text{Paths}(\text{Paths}(G')) \xrightarrow{\mu_{G'}} \text{Paths}(G') \tag{4.14}$$

This says that given a function from arrows in G to paths in G', a path in G becomes a path of paths in G', which can be concatenated to a path in G'. This simply and precisely spells out our intuition.

Definition 4.4.1.2 (Schema morphism)**.** Let $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ be graphs, and let $\mathcal{C} = (G, \simeq_G)$ and $\mathcal{C}' = (G', \simeq_{G'})$ be schemas. A *schema morphism F from $\mathcal{C}$ to $\mathcal{D}$*, denoted $F\colon \mathcal{C} \to \mathcal{D}$ is a graph homomorphism [22]

$$F\colon G \to \text{Paths}(G')$$

that satisfies the following condition for any paths p and q in G:

$$\text{if} \quad p \simeq_G q \quad \text{then} \quad \text{Paths}_F(p) \simeq_{G'} \text{Paths}_F(q). \tag{4.15}$$

Two schema morphisms $E, F\colon \mathcal{C} \to \mathcal{C}'$ are considered identical if they agree on vertices (i.e. $E_0 = F_0$) and if, for every arrow f in G, there is a path equivalence in G'

$$E_1(f) \simeq_{G'} F_1(f).$$

[22]By Definition 3.3.3.1, a graph homomorphism $F\colon G \to \text{Paths}(G')$ will consist of a vertex part $F_0\colon V \to V'$ and an arrows part $F_1\colon E \to \text{Path}(G')$. See also Definition 3.3.2.1.

We now define the *category of schemas*, denoted **Sch**, to be the category whose objects are schemas as in Definition 3.5.2.6 and whose morphisms are schema morphisms defined as above. The identity morphism on schema $\mathcal{C} = (G, \simeq_G)$ is the schema morphism $\mathrm{id}_{\mathcal{C}} := \eta_G \colon G \to \mathrm{Paths}(G)$ as defined in Equation (4.13). We need only understand how to compose schema morphisms $F \colon \mathcal{C} \to \mathcal{C}'$ and $F' \colon \mathcal{C}' \to \mathcal{C}''$. On objects their composition is obvious. Given an arrow in $\mathcal{C}$, it is sent to a path in $\mathcal{C}'$; each arrow in that path is sent to a path in $\mathcal{C}''$. We then have a path of paths which we can concatenate (via $\mu_{G''} \colon \mathrm{Paths}(\mathrm{Paths}(G'')) \to \mathrm{Paths}(G'')$ as in 4.13) to get a path in $\mathcal{C}''$ as desired.

Slogan 4.4.1.3.

> " *A schema morphism sends vertices to vertices, arrows to paths, and path equivalences to path equivalences.* "

Example 4.4.1.4. Let $[2]$ be the linear order graph of length 2, pictured to the left, and let $\mathcal{C}$ denote the schema pictured to the right below:

$$[2] := \begin{tikzcd} \overset{0}{\bullet} \arrow[r, "f_1"] & \overset{1}{\bullet} \arrow[r, "f_2"] & \overset{2}{\bullet} \end{tikzcd} \qquad \mathcal{C} := \begin{tikzcd} \overset{a}{\bullet} \arrow[r, "g"] \arrow[dr, "i"'] & \overset{b}{\bullet} \arrow[d, "h"] \\ & \overset{c}{\bullet} \end{tikzcd}$$

We impose on $\mathcal{C}$ the path equivalence declaration $[g, h] \simeq [i]$ and show that in this case $\mathcal{C}$ and $[2]$ are isomorphic in **Sch**. We have a schema morphism $F \colon [2] \to \mathcal{C}$ sending $0 \mapsto a, 1 \mapsto b, 2 \mapsto c$, and sending each arrow in $[2]$ to an arrow in $\mathcal{C}$. And we have a schema morphism $F' \colon \mathcal{C} \to [2]$ which reverses this mapping on vertices; note that F' must send the arrow i in $\mathcal{C}$ to the path $[f_1, f_2]$ in $[2]$, which is ok! The roundtrip $F' \circ F \colon [2] \to [2]$ is identity. The roundtrip $F \circ F' \colon \mathcal{C} \to \mathcal{C}$ may look like it's not the identity; indeed it sends vertices to themselves but it sends i to the path $[g, h]$. But according to Definition 4.4.1.2, this schema morphism is considered identical to $\mathrm{id}_{\mathcal{C}}$ because there is a path equivalence $\mathrm{id}_{\mathcal{C}}(i) = [i] \simeq [g, h] = F \circ F'(i)$.

Exercise 4.4.1.5. Consider the schema $[2]$ and the schema $\mathcal{C}$ pictured above, except where this time we *do not* impose any path equivalence declarations on $\mathcal{C}$, so $[g, h] \not\simeq [i]$ in our current version of $\mathcal{C}$.

a.) How many schema morphisms are there $[2] \to \mathcal{C}$ that send 0 to a?

b.) How many schema morphisms are there $\mathcal{C} \to [2]$ that send a to 0?

$\Diamond$

Exercise 4.4.1.6. Consider the graph $\mathcal{L}oop$ pictured below

$$\mathcal{L}oop := \begin{tikzcd} \overset{s}{\bullet} \arrow[loop, "f"] \end{tikzcd}$$

and for any natural number n, let $\mathcal{L}_n$ denote the schema $(\mathcal{L}oop, \simeq_n)$ where $\simeq_n$ is the PED $f^{n+1} \simeq f^n$. This is the "finite hierarchy" schema of Example 3.5.2.11. Let $\underline{1}$ denote the graph with one vertex and no arrows; consider it as a schema.

a.) Is $\underline{1}$ isomorphic to $\mathcal{L}_1$ in **Sch**?

b.) Is it isomorphic to any (other) $\mathcal{L}_n$?

$\lozenge$

Exercise 4.4.1.7. Let $\mathcal{L}oop$ and $\mathcal{L}_n$ be the schemas defined in Exercise 4.4.1.6.

a.) What is the cardinality of the set $\mathrm{Hom}_{\mathbf{Sch}}(\mathcal{L}_3, \mathcal{L}_5)$?

b.) What is the cardinality of the set $\mathrm{Hom}_{\mathbf{Sch}}(\mathcal{L}_5, \mathcal{L}_3)$? Hint: the cardinality of the set $\mathrm{Hom}_{\mathbf{Sch}}(\mathcal{L}_4, \mathcal{L}_9)$ is 8.

$\lozenge$

4.4.2 Proving the equivalence

Construction 4.4.2.1 (From schema to category). We will define a functor $L\colon \mathbf{Sch} \to \mathbf{Cat}$. Let $\mathcal{C} = (G, \simeq)$ be a categorical schema, where $G = (V, A, src, tgt)$. Define $L(\mathcal{C})$ to be the category with $\mathrm{Ob}(L(\mathcal{C})) = V$, and with $\mathrm{Hom}_{L(\mathcal{C})}(v_1, v_2) := \mathrm{Path}_G(v, w)/\simeq$, i.e. the set of paths in G, modulo the path equivalence relation for $\mathcal{C}$. The composition of morphisms is defined by concatenation of paths, and Lemma 3.5.2.5 ensures that such composition is well-defined. We have thus defined L on objects of **Sch**.

Given a schema morphism $F\colon \mathcal{C} \to \mathcal{C}'$, where $\mathcal{C}' = (G', \simeq')$, we need to produce a functor $L(F)\colon L(\mathcal{C}) \to L(\mathcal{C}')$. The objects of $L(\mathcal{C})$ and $L(\mathcal{C}')$ are the vertices of G and G' respectively, and F provides the necessary function on objects. Diagram (4.14) provides a function $\mathrm{Paths}_F\colon \mathrm{Paths}(G) \to \mathrm{Paths}(G')$ will provide the requisite function for morphisms.

A morphism in $L(\mathcal{C})$ is an equivalence class of paths in $\mathcal{C}$. For any representative path $p \in \mathrm{Paths}(G)$, we have $\mathrm{Paths}_F(p) \in \mathrm{Paths}(G')$, and if $p \simeq q$ then $\mathrm{Paths}_F(p) \simeq' \mathrm{Paths}_F(q)$ by condition 4.15. Thus Paths_F indeed provides us with a function $\mathrm{Hom}_{L(\mathcal{C})} \to \mathrm{Hom}_{L(\mathcal{C}')}$. This defines L on morphisms in **Sch**. It is clear that L preserves composition and identities, so it is a functor.

Construction 4.4.2.2 (From category to schema). We will define a functor $R\colon \mathbf{Cat} \to \mathbf{Sch}$. Let $\mathcal{C} = (\mathrm{Ob}(\mathcal{C}), \mathrm{Hom}_{\mathcal{C}}, dom, cod, \mathrm{ids}, \circ)$ be a category (see Exercise 4.1.1.23). Let $R(\mathcal{C}) = (G, \simeq)$ where G is the graph

$$G = (\mathrm{Ob}(\mathcal{C}), \mathrm{Hom}_{\mathcal{C}}, dom, cod),$$

and with $\simeq$ defined as the congruence generated by the following path equivalence declarations: for any composable sequence of morphisms $f_1, f_2, \ldots, f_n$ (with $dom(f_{i+1}) = cod(f_i)$ for each $1 \leqslant i \leqslant n-1$) we put

$$[f_1, f_2, \ldots, f_n] \simeq [f_n \circ \cdots \circ f_2 \circ f_1]. \tag{4.16}$$

This defines R on objects of **Cat**.

A functor $F\colon \mathcal{C} \to \mathcal{D}$ induces a schema morphism $R(F)\colon R(\mathcal{C}) \to R(\mathcal{D})$, because vertices are sent to vertices, arrows are sent to arrows (as paths of length 1), and path equivalence is preserved by (5.14) and the fact that F preserves the composition formula. This defines R on morphisms in **Cat**. It is clear that R preserves compositions, so it is a functor.

Theorem 4.4.2.3. *The functors*

$$L\colon \mathbf{Sch} \rightleftarrows \mathbf{Cat}\colon R$$

are mutually inverse equivalences of categories.

Sketch of proof. It is clear that there is a natural isomorphism $\alpha\colon \mathrm{id}_{\mathbf{Cat}} \xrightarrow{\cong} L \circ R$; i.e. for any category $\mathcal{C}$, there is an isomorphism $\mathcal{C} \cong L(R(\mathcal{C}))$.

Before giving an isomorphism $\beta\colon \mathrm{id}_{\mathbf{Sch}} \xrightarrow{\cong} R \circ L$, we briefly describe $R(L(\mathcal{S})) =: (G', \simeq')$ for a schema $\mathcal{S} = (G, \simeq)$. Write $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$. On vertices we have $V = V'$. On arrows we have $A' = \mathrm{Path}_G / \simeq$. The congruence $\simeq'$ for $R(L(\mathcal{S}))$ is imposed in (5.14). Under $\simeq'$, every path of paths in G is made equivalent to its concatenation, considered as a path of length 1 in G'.

There is a natural transformation $\beta\colon \mathrm{id}_{\mathbf{Sch}} \to R \circ L$ whose $\mathcal{S}$-component sends each arrow in G to a certain path of length 1 in G'. We need to see that $\beta_{\mathcal{S}}$ has an inverse. But this is straightforward: every arrow f in $R \circ L(\mathcal{S})$ is an equivalence class of paths in $\mathcal{S}$; choose any one and send f there; by Definition 4.4.1.2 any other choice will give the identical morphism of schemas. It is easy to show that the roundtrips are identities (again up to the notion of identity given in Definition 4.4.1.2).

□

4.5 Limits and colimits

Limits and colimits are universal constructions, meaning they represent certain ideals of behavior in a category. When it comes to sets that map to A and B, the $(A \times B)$-grid is ideal—it projects on to both A and B as straightforwardly as possible. When it comes to sets that can interpret the elements of both A and B, the disjoint union $A \sqcup B$ is ideal—it includes both A and B without confusion or superfluity. These are limits and colimits in **Set**. Limits and colimits exist in other categories as well.

Limits in a preorder are meets, colimits in a preorder are joins. Limits and colimits also exist for database instances and monoid actions, allowing us to discuss for example the product or union of different state machines. Limits and colimits exist for spaces, giving rise to products and unions, as well as quotients.

Limits and colimits do not exist in every category; when $\mathcal{C}$ is complete with respect to limits (or colimits), these limits always seem to mean something valuable to human intuition. For example, when a subject has already been studied for a long time before category theory came around, it often turns out that classically interesting constructions in the subject correspond to limits and colimits in its categorification $\mathcal{C}$. For example products, unions, equivalence relations, etc. are classical ideas in set theory that are naturally captured by limits and colimits in **Set**.

4.5.1 Products and coproducts in a category

In Sections 2.4, we discussed products and coproducts in the category **Set** of sets. Now we discuss the same notions in an arbitrary category. For both products and coproducts we will begin with examples and then write down the general concept, but we'll work on products first.

4.5.1.1 Products

The product of two sets is a grid, which projects down onto each of the two sets. This is good intuition for products in general.

Example 4.5.1.2. Given two preorders, $\mathcal{X}_1 := (X_1, \leqslant_1)$ and $\mathcal{X}_2 := (X_2, \leqslant_2)$, we can take their product and get a new preorder $\mathcal{X}_1 \times \mathcal{X}_2$. Both $\mathcal{X}_1$ and $\mathcal{X}_2$ have underlying sets (namely X_1 and X_2), so we might hope that the underlying set of $\mathcal{X}_1 \times \mathcal{X}_2$ is the set $X_1 \times X_2$ of ordered pairs, and this turns out to be true. We have a notion of less-than on $\mathcal{X}_1$ and we have a notion of less-than on $\mathcal{X}_2$; we need to construct a notion of less-than on $\mathcal{X}_1 \times \mathcal{X}_2$. So, given two ordered pairs (x_1, x_2) and (x_1', x_2'), when should we say that $(x_1, x_2) \leqslant_{1,2} (x_1', x_2')$ holds? The obvious guess is to say that it holds iff both $x_1 \leqslant_1 x_1'$ and $x_2 \leqslant_2 x_2'$ hold, and this works:

$$\mathcal{X}_1 \times \mathcal{X}_2 := (X_1 \times X_2, \leqslant_{1,2})$$

Note that the projection functions $X_1 \times X_2 \to X_1$ and $X_1 \times X_2 \to X_2$ induce morphisms of preorders. That is, if $(x_1, x_2) \leqslant_{1,2} (x_1', x_2')$ then in particular $x_1 \leqslant x_1'$. So we have preorder morphisms

$$\begin{tikzcd} & \mathcal{X}_1 \times \mathcal{X}_2 \arrow[dl] \arrow[dr] & \\ \mathcal{X}_1 & & \mathcal{X}_2 \end{tikzcd}$$

Exercise 4.5.1.3. Suppose that you have a partial order $(S, \leqslant_S)$ on songs (so you know some songs are preferable to others but sometimes you can't compare). And suppose you have a partial order $(A, \leqslant_A)$ on pieces of art. You're about to be given a pair (s, a) including a song and a piece of art. Does the product partial order $\mathcal{S} \times \mathcal{A}$ provide a reasonable guess for your preferences on pairs? ◊

Exercise 4.5.1.4. Consider the partial order $\leqslant$ on $\mathbb{N}$ given by standard "less-than-or-equal-to", so $5 \leqslant 9$ etc. And consider another partial order, `divides` on $\mathbb{N}$, where a `divides` b if "a goes into b evenly", i.e. if there exists $n \in \mathbb{N}$ such that $a * n = b$, so 5 `divides` 35. If we call the product order $(X, \preceq) := (\mathbb{N}, \leqslant) \times (\mathbb{N}, \texttt{divides})$, which of the following are true:

$$(2,4) \preceq (3,4)? \qquad (2,4) \preceq (3,5)? \qquad (2,4) \preceq (8,0)? \qquad (2,4) \preceq (0,0)?$$

◊

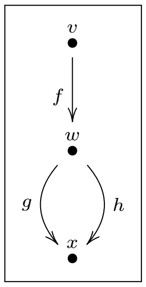
Example 4.5.1.5. Given two graphs $G_1 = (V_1, A_1, src_1, tgt_1)$ and $G_2 = (V_2, A_2, src_2, tgt_2)$, we can take their product and get a new graph $G_1 \times G_2$. The vertices will be the grid of vertices $V_1 \times V_2$, so each vertex in $G_1 \times G_2$ is labeled by a pair of vertices, one from G_1 and one from G_2. When should an arrow connect (v_1, v_2) to (v_1', v_2')? Whenever we can find an arrow in G_1 connecting v_1 to v_1' and we can find an arrow in G_2 connecting v_2 to v_2'. It turns out there is a simple formula for the set of arrows in $G_1 \times G_2$, namely $A_1 \times A_2$.

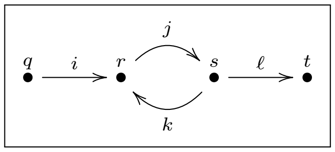
Let's write $G := G_1 \times G_2$ and say $G = (V, A, src, tgt)$. We now know that $V = V_1 \times V_2$ and $A = A_1 \times A_2$. What should the source and target functions $A \to V$ be? Given a function $src_1 \colon A_1 \to V_1$ and a function $src_2 \colon A_2 \to V_2$, the universal property of products in **Set** (Lemma 2.4.1.10 or better Example 2.4.1.16) provides a unique function

$$src := src_1 \times src_2 \colon A_1 \times A_2 \to V_1 \times V_2$$

Namely the source of arrow (a_1, a_2) will be the vertex $(src_1(a_1), src_2(a_2))$. Similarly we have a ready-made choice of target function $tgt = tgt_1 \times tgt_2$. We have now defined the product graph.

Here's a concrete example. Let I and J be as drawn below:

$I :=$ 

$J :=$ 

Arrow (I)		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x

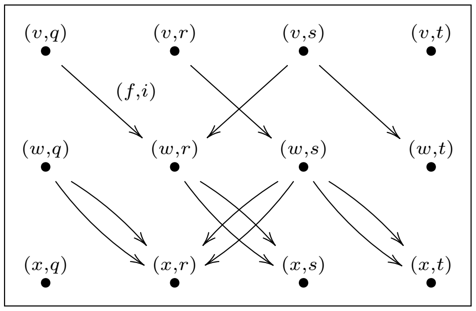
Vertex (I)
ID
v
w
x

Arrow (J)		
ID	**src**	**tgt**
i	q	r
j	r	s
k	s	r
ℓ	s	t

Vertex (J)
ID
q
r
s
t

The product $I \times J$ drawn below has, as expected $3*4 = 12$ vertices and $3*4 = 12$ arrows:

$I \times J :=$



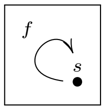
Arrow $(I \times J)$		
ID	**src**	**tgt**
(f, i)	(v, q)	(w, r)
(f, j)	(v, r)	(w, s)
(f, k)	(v, s)	(w, r)
(f, ℓ)	(v, s)	(w, t)
(g, i)	(w, q)	(x, r)
(g, j)	(w, r)	(x, s)
(g, k)	(w, s)	(x, r)
(g, ℓ)	(w, s)	(x, t)
(h, i)	(w, q)	(x, r)
(h, j)	(w, r)	(x, s)
(h, k)	(w, s)	(x, r)
(h, ℓ)	(w, s)	(x, t)

Vertex $(I \times J)$
ID
(v, q)
(v, r)
(v, s)
(v, t)
(w, q)
(w, r)
(w, s)
(w, t)
(x, q)
(x, r)
(x, s)
(x, t)

Here is the most important thing to notice. Look at the `Arrow` table for $I \times J$, and for each ordered pair, look only at the second entry in all three columns; you will see something that matches with the `Arrow` table for J. Do the same for I, and again you'll see a perfect match. These "matchings" are readily-visible graph homomorphisms $I \times J \to I$ and $I \times J \to J$ in **Grph**.

Exercise 4.5.1.6. Let $[1] = \boxed{\overset{0}{\bullet} \xrightarrow{f} \overset{1}{\bullet}}$ be the linear order graph of length 1 and let $P = \text{Paths}([1])$ be its paths-graph, as in Example 4.1.2.22 (so P should have three arrows and two vertices). Draw the graph $P \times P$. ◊

Exercise 4.5.1.7. Recall from Example 3.5.2.9 that a discrete dynamical system (DDS) is a set s together with a function $f \colon s \to s$. By now it should be clear that if

$\mathcal{L}oop :=$  is the loop schema, then a DDS is simply an instance (a functor) $I\colon \mathcal{L}oop \to \mathbf{Set}$. We have not yet discussed products of DDS's, but perhaps you can guess how they should work. For example, consider the instances $I, J\colon \mathcal{L}oop \to \mathbf{Set}$ tabulated below:

s (I)	
ID	**f**
A	C
B	C
C	C

s (J)	
ID	**f**
x	y
y	x
z	z

a.) Make a guess and tabulate $I \times J$. Then draw it.[23]

b.) Recall the notion of natural transformations between functors (see Example 4.3.3.5), which in the case of functors $\mathcal{L}oop \to \mathbf{Set}$ are the morphisms of instances. Do you see clearly that there is a morphism of instances $I \times J \to I$ and $I \times J \to J$? Just check that if you look only at the left-hand coordinates in your $I \times J$, you see something compatible with I.

◊

In every case above, what's most important to recognize is that there are projection maps $I \times J \to I$ and $I \times J \to J$, and that the construction of $I \times J$ seems as straightforward as possible, subject to having these projections. It is time to give the definition.

Definition 4.5.1.8. Let $\mathcal{C}$ be a category and let $X, Y \in \mathrm{Ob}(\mathcal{C})$ be objects. A *span on X and Y* consists of three constituents (Z, p, q), where $Z \in \mathrm{Ob}(\mathcal{C})$ is an object, and where $p\colon Z \to X$ and $q\colon Z \to Y$ are morphisms in $\mathcal{C}$.

$$\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\ X & & Y \end{tikzcd}$$

A *product of X and Y* is a span $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$, [24] such that for any other span $X \xleftarrow{p} Z \xrightarrow{q} Y$ there *exists a unique* morphism $t_{p,q}\colon Z \to X \times Y$ such that the diagram below commutes:

$$\begin{tikzcd} & X \times Y \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & \\ X & & Y \\ & Z \arrow[ul, "p"] \arrow[ur, "q"'] \arrow[uu, dashed, "t_{p,q}"] & \end{tikzcd}$$

Remark 4.5.1.9. Definition 4.5.1.8 endows the product of two objects with something known as a *universal property*. It says that a product of two objects X and Y maps to

[23]The result is not necessarily inspiring, but at least computing it is straightforward.
[24]The names $X \times Y$ and π_1, π_2 are not mathematically important, they are pedagogically suggestive.

those two objects, and serves as a gateway for all who do the same. "None shall map to X and Y except through me!" This grandiose property is held by products in all the various categories we have discussed so far. It is what I meant when I said things like "$X \times Y$ maps to both X and Y and does so as straightforwardly as possible". The grid of dots obtained as the product of two sets has such a property, as was shown in Example 2.4.1.11.

Example 4.5.1.10. In Example 4.5.1.2 we discussed products of preorders. In this example we will discuss products in an individual preorder. That is, by Proposition 4.2.1.17, there is a functor **PrO** $\to$ **Cat** that realizes every preorder as a category. If $\mathcal{P} = (P, \leqslant)$ is a preorder, what are products in $\mathcal{P}$? Given two objects $a, b \in \text{Ob}(\mathcal{P})$ we first consider spans on a and b, i.e. $a \leftarrow z \rightarrow b$. That would be some z such that $z \leqslant a$ and $z \leqslant b$. The product will be such a span $a \geqslant a \times b \leqslant b$, but such that every other spanning object z is less than or equal to $a \times b$. In other words $a \times b$ is as big as possible subject to the condition of being less than a and less than b. This is precisely the meet of a and b (see Definition 3.4.2.1).

Example 4.5.1.11. Note that the product of two objects in a category $\mathcal{C}$ may not exist. Let's return to preorders to see this phenomenon.

Consider the set $\mathbb{R}^2$, and say that $(x_1, y_1) \leqslant (x_2, y_2)$ if there exists $\ell \geqslant 1$ such that $x_1\ell = x_2$ and $y_1\ell = y_2$; in other words, point p is less than point q if, in order to travel from q to the origin along a straight line, one must pass through p along the way. [25] We have given a perfectly good partial order, but $p := (1, 0)$ and $q := (0, 1)$ do not have a product. Indeed, it would have to be a non-zero point that was on the same line-through-the origin as p and the same line-through-the-origin as q, of which there are none.

Example 4.5.1.12. Note that there can be more than one product of two objects in a category $\mathcal{C}$, but that any two choices will be canonically isomorphic. Let's return once more to preorders to see this phenomenon.

Consider the set $\mathbb{R}^2$ and say that $(x_1, y_1) \leqslant (x_2, y_2)$ if $x_1^2 + y_1^2 \leqslant x_2^2 + y_2^2$, in other words if the former is on a smaller 0-circle (by which I mean "circle centered at the origin") than the latter is.

For any two points p, q there will be lots of points that serve as products: anything on the smaller of their two 0-circles will suffice. Given any two points a, b on this smaller circle, we will have a unique isomorphism $a \cong b$ because $a \leqslant b$ and $b \leqslant a$ and all morphisms are unique in a preorder.

Exercise 4.5.1.13. Consider the preorder $\mathcal{P}$ of cards in a deck, shown in Example 3.4.1.3; it is not the entire story of cards in a deck, but take it to be so. In other words, be like a computer and take what's there at face value. Consider the preorder $\mathcal{P}$ as a category (by way of the functor **PrO** $\to$ **Cat**).

a.) For each of the following pairs, what is their product in $\mathcal{P}$ (if it exists)?

$$\ulcorner\text{a diamond}\urcorner \times \ulcorner\text{a heart}\urcorner\,? \qquad \ulcorner\text{a queen}\urcorner \times \ulcorner\text{a black card}\urcorner\,?$$
$$\ulcorner\text{a card}\urcorner \times \ulcorner\text{a red card}\urcorner\,? \qquad \ulcorner\text{a face card}\urcorner \times \ulcorner\text{a black card}\urcorner\,?$$

b.) How would these answers differ if $\mathcal{P}$ was completed to the "whole story" partial order classifying cards in a deck?

◊

[25] Note that $(0, 0)$ is not related to anything else.

Exercise 4.5.1.14. Let X be a set, and consider it as a discrete category. Given two objects $x, y \in \text{Ob}(X)$, under what conditions will there exist a product $x \times y$? ◊

Exercise 4.5.1.15. Let $f\colon \mathbb{R} \to \mathbb{R}$ be a function, like you would see in 6th grade (maybe $f(x) = x + 7$). A typical thing to do is to graph f as a curve running through the plane $\mathbb{R}^2 := \mathbb{R} \times \mathbb{R}$. This curve can be understood as a function $F\colon \mathbb{R} \to \mathbb{R}^2$.

a.) Given some $x \in \mathbb{R}$, what are the coordinates of $F(x) \in \mathbb{R}^2$?

b.) Obtain $F\colon \mathbb{R} \to \mathbb{R}^2$ using the universal property given in Definition 4.5.1.8.

◊

Exercise 4.5.1.16. Consider the preorder $(\mathbb{N}, \texttt{divides})$, discussed in Exercise 4.5.1.4, where e.g. $5 \leqslant 15$ but $5 \nleqslant 6$.

a.) What is the product of 9 and 12 in this category?

b.) Is there a standard name for products in this category?

◊

Example 4.5.1.17. All products exist in the category **Cat**. Given two categories $\mathcal{C}$ and $\mathcal{D}$, there is a product category $\mathcal{C} \times \mathcal{D}$. We have $\text{Ob}(\mathcal{C} \times \mathcal{D}) = \text{Ob}(\mathcal{C}) \times \text{Ob}(\mathcal{D})$ and for any two objects (c, d) and (c', d'), we have

$$\text{Hom}_{\mathcal{C}\times\mathcal{D}}((c,d),(c',d')) = \text{Hom}_{\mathcal{C}}(c,c') \times \text{Hom}_{\mathcal{C}}(d,d').$$

The composition formula is "obvious".

Let $[1] \in \text{Ob}(\textbf{Cat})$ denote the linear order category of length 1, drawn

$$[1] := \boxed{\begin{tikzcd} \overset{0}{\bullet} \arrow[r, "f"] & \overset{1}{\bullet} \end{tikzcd}}$$

As a schema it has one arrow, but as a category it has three morphisms. So we expect $[1] \times [1]$ to have 9 morphisms, and that's true. In fact, $[1] \times [1]$ looks like a commutative square:

$$\begin{tikzcd} \overset{(0,0)}{\bullet} \arrow[r, "\text{id}_0 \times f"] \arrow[d, "f \times \text{id}_0"'] & \overset{(0,1)}{\bullet} \arrow[d, "f \times \text{id}_1"] \\ \overset{(1,0)}{\bullet} \arrow[r, "\text{id}_1 \times f"'] & \overset{(1,1)}{\bullet} \end{tikzcd} \tag{4.17}$$

We see only four morphisms here, but there are also four identities and one morphism $(0, 0) \to (1, 1)$ given by composition of either direction. It is a minor miracle that the categorical product somehow "knows" that this square should commute; however, this is not the mere preference of man but instead the dictate of God! By which I mean, this follows rigorously from the definitions we already gave of **Cat** and products.

4.5.1.18 Coproducts

The coproduct of two sets is their disjoint union, which includes non-overlapping copies of each of the two sets. This is good intuition for coproducts in general.

Example 4.5.1.19. Given two preorders, $\mathcal{X}_1 := (X_1, \leqslant_1)$ and $\mathcal{X}_2 := (X_2, \leqslant_2)$, we can take their coproduct and get a new preorder $\mathcal{X}_1 \sqcup \mathcal{X}_2$. Both $\mathcal{X}_1$ and $\mathcal{X}_2$ have underlying sets (namely X_1 and X_2), so we might hope that the underlying set of $\mathcal{X}_1 \times \mathcal{X}_2$ is the disjoint union $X_1 \sqcup X_2$, and that turns out to be true. We have a notion of less-than on $\mathcal{X}_1$ and we have a notion of less-than on $\mathcal{X}_2$.

Given an element $x \in X_1 \sqcup X_2$ and an element $x' \in X_1 \sqcup X_2$, how can we use $\leqslant_1$ and $\leqslant_2$ to compare x_1 and x_2? The relation $\leqslant_1$ only knows how to compare elements of X_1 and the relation $\leqslant_2$ only knows how to compare elements of X_2. But x and x' may come from different homes; e.g. $x \in X_1$ and $x' \in X_2$, in which case neither $\leqslant_1$ nor $\leqslant_2$ gives any clue about which should be bigger.

So when should we say that $x \leqslant_{1\sqcup 2} x'$ holds? The obvious guess is to say that x is less than x' iff somebody says it is; that is, if both x and x' are from the same home and the local ordering has $x \leqslant x'$. To be precise, we say $x \leqslant_{1\sqcup 2} x'$ if and only if either one of the following conditions hold:

- $x \in X_1$ and $x' \in X_1$ and $x \leqslant_1 x'$, or
- $x \in X_2$ and $x' \in X_2$ and $x \leqslant_2 x'$.

With $\leqslant_{1\sqcup 2}$ so defined, one checks that it is not only a preorder, but that it serves as a coproduct of $\mathcal{X}_1$ and $\mathcal{X}_2$,

$$\mathcal{X}_1 \sqcup \mathcal{X}_2 := (X_1 \sqcup X_2, \leqslant_{1\sqcup 2}).$$

Note that the inclusion functions $X_1 \to X_1 \sqcup X_2$ and $X_2 \to X_1 \sqcup X_2$ induce morphisms of preorders. That is, if $x, x' \in X_1$ are elements such that $x \leqslant_1 x'$ in $\mathcal{X}_1$ then the same will hold in $\mathcal{X}_1 \sqcup \mathcal{X}_2$. So we have preorder morphisms

$$\begin{tikzcd}
& \mathcal{X}_1 \sqcup \mathcal{X}_2 & \\
\mathcal{X}_1 \arrow[ur] & & \mathcal{X}_2 \arrow[ul]
\end{tikzcd}$$

Exercise 4.5.1.20. Suppose that you have a partial order $\mathcal{A} := (A, \leqslant_A)$ on apples (so you know some apples are preferable to others but sometimes you can't compare). And suppose you have a partial order $\mathcal{O} := (O, \leqslant_O)$ on oranges. You're about to be given two pieces of fruit from a basket of apples and oranges. Is the coproduct partial order $\mathcal{A} \sqcup \mathcal{O}$ a reasonable guess for your preferences, or does it seem biased? ◊

Example 4.5.1.21. Given two graphs $G_1 = (V_1, A_1, src_1, tgt_1)$ and $G_2 = (V_2, A_2, src_2, tgt_2)$, we can take their coproduct and get a new graph $G_1 \sqcup G_2$. The vertices will be the disjoint union of vertices $V_1 \sqcup V_2$, so each vertex in $G_1 \sqcup G_2$ is labeled either by a vertex in G_1 or by one in G_2 (and if any labels are shared, then something must be done to differentiate them). When should an arrow connect v to v'? Whenever both are from the same component (i.e. either $v, v' \in V_1$ or $v, v' \in V_2$) and we can find an arrow connecting them in that component. It turns out there is a simple formula for the set of arrows in $G_1 \sqcup G_2$, namely $A_1 \sqcup A_2$.

Let's write $G := G_1 \sqcup G_2$ and say $G = (V, A, src, tgt)$. We now know that $V = V_1 \sqcup V_2$ and $A = A_1 \sqcup A_2$. What should the source and target functions $A \to V$ be? Given a function $src_1 \colon A_1 \to V_1$ and a function $src_2 \colon A_2 \to V_2$, the universal property of coproducts in **Set** can be used to specify a unique function

$$src := src_1 \sqcup src_2 \colon A_1 \sqcup A_2 \to V_1 \sqcup V_2.$$

Namely for any arrow $a \in A$, we know either $a \in A_1$ or $a \in A_2$ (and not both), so the source of a will be the vertex $src_1(a)$ if $a \in A_1$ and $src_2(a)$ if $a \in A_2$. Similarly we have a ready-made choice of target function $tgt = tgt_1 \sqcup tgt_2$. We have now defined the coproduct graph.

Here's a real example. Let I and J be as in Example 4.3.3.5, drawn below:

$$I := \boxed{\begin{tikzcd} \overset{v}{\bullet} \arrow[d, "f"'] \\ \overset{w}{\bullet} \arrow[d, bend right, "g"'] \arrow[d, bend left, "h"] \\ \overset{x}{\bullet} \end{tikzcd}} \qquad J := \boxed{\begin{tikzcd} \overset{q}{\bullet} \arrow[r, "i"] & \overset{r}{\bullet} \arrow[r, bend left, "j"] & \overset{s}{\bullet} \arrow[l, bend left, "k"] \arrow[r, "\ell"] & \overset{t}{\bullet} \\ & & & \overset{u}{\bullet} \end{tikzcd}}$$

Arrow (I)		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x

Vertex (I)
ID
v
w
x

Arrow (J)		
ID	**src**	**tgt**
i	q	r
j	r	s
k	s	r
ℓ	s	t

Vertex (J)
ID
q
r
s
t
u

The coproduct $I \sqcup J$ drawn below has, as expected $3 + 5 = 8$ vertices and $3 + 4 = 7$ arrows:

$$I \sqcup J := \boxed{\begin{tikzcd} \overset{v}{\bullet} \arrow[d, "f"'] & & & & \\ \overset{w}{\bullet} \arrow[d, bend right, "g"'] \arrow[d, bend left, "h"] & \overset{q}{\bullet} \arrow[r, "i"] & \overset{r}{\bullet} \arrow[r, bend left, "j"] & \overset{s}{\bullet} \arrow[l, bend left, "k"] \arrow[r, "\ell"] & \overset{t}{\bullet} \\ \overset{x}{\bullet} & & & & \overset{u}{\bullet} \end{tikzcd}}$$

Arrow $(I \sqcup J)$		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x
i	q	r
j	r	s
k	s	r
ℓ	s	t

Vertex $(I \sqcup J)$
ID
v
w
x
q
r
s
t
u

Here is the most important thing to notice. Look at the `Arrow` table I and notice that there is a way to send each row to a row in $I \sqcup J$, such that all the foreign keys match. Similarly in the arrow table and the two vertex tables for J. These "matchings" are readily-visible graph homomorphisms $I \to I \sqcup J$ and $J \to I \sqcup J$ in **Grph**.

Exercise 4.5.1.22. Recall from Example 3.5.2.9 that a discrete dynamical system (DDS) is a set s together with a function $f: s \to s$; if

$$\mathcal{L}oop := \boxed{\begin{tikzcd} \overset{s}{\bullet} \arrow[loop above, "f"] \end{tikzcd}}$$

is the loop schema, then a DDS is simply an instance (a functor) $I: \mathcal{L}oop \to \mathbf{Set}$. We have not yet discussed coproducts of DDS's, but perhaps you can guess how they should work. For example, consider the instances $I, J\colon \mathcal{L}oop \to \mathbf{Set}$ tabulated below:

s (I)	
ID	**f**
A	C
B	C
C	C

s (J)	
ID	**f**
x	y
y	x
z	z

Make a guess and tabulate $I \sqcup J$. Then draw it. $\Diamond$

In every case above (preorders, graphs, DDSs), what's most important to recognize is that there are inclusion maps $I \to I \sqcup J$ and $J \to I \sqcup J$, and that the construction of $I \sqcup J$ seems as straightforward as possible, subject to having these inclusions. It is time to give the definition.

Definition 4.5.1.23. Let $\mathcal{C}$ be a category and let $X, Y \in \mathrm{Ob}(\mathcal{C})$ be objects. A *cospan on X and Y* consists of three constituents (Z, i, j), where $Z \in \mathrm{Ob}(\mathcal{C})$ is an object, and where $i\colon X \to Z$ and $j\colon Y \to Z$ are morphisms in $\mathcal{C}$.

$$\begin{tikzcd}
 & Z & \\
X \arrow[ur, "i"] & & Y \arrow[ul, "j"']
\end{tikzcd}$$

A *coproduct of X and Y* is a cospan $X \xrightarrow{\iota_1} X \sqcup Y \xleftarrow{\iota_2} Y$, [26] such that for any other cospan $X \xrightarrow{i} Z \xleftarrow{j} Y$ there *exists a unique* morphism $s_{i,j}\colon X \sqcup Y \to Z$ such that the diagram below commutes:

$$\begin{tikzcd}
 & X \sqcup Y \arrow[dd, dashed, "s_{i,j}"'] & \\
X \arrow[ur, "\iota_1"] \arrow[dr, "i"'] & & Y \arrow[ul, "\iota_2"'] \arrow[dl, "j"] \\
 & Z &
\end{tikzcd}$$

Remark 4.5.1.24. Definition 4.5.1.8 endows the coproduct of two objects with a *universal property.* It says that a coproduct of two objects X and Y receives maps from those two objects, and serves as a gateway for all who do the same. "None shall receive maps from X and Y except through me!" This grandiose property is held by all the coproducts we have discussed so far. It is what I meant when I said things like "$X \sqcup Y$ receives maps from both X and Y and does so as straightforwardly as possible". The disjoint union of dots obtained as the coproduct of two sets has such a property, as can be seen by thinking about Example 2.4.2.5.

Example 4.5.1.25. By Proposition 4.2.1.17, there is a functor $\mathbf{PrO} \to \mathbf{Cat}$ that realizes every preorder as a category. If $\mathcal{P} = (P, \leqslant)$ is a preorder, what are coproducts in $\mathcal{P}$? Given two objects $a, b \in \mathrm{Ob}(\mathcal{P})$ we first consider cospans on a and b, i.e. $a \to z \leftarrow b$.

[26]The names $X \sqcup Y$ and ι_1, ι_2 are not mathematically important, they are pedagogically suggestive.

A cospan of a and b is any z such that $a \leqslant z$ and $b \leqslant z$. The coproduct will be such a cospan $a \leqslant a \sqcup b \geqslant b$, but such that every other cospanning object z is greater than or equal to $a \sqcup b$. In other words $a \sqcup b$ is as small as possible subject to the condition of being bigger than a and bigger than b. This is precisely the join of a and b (see Definition 3.4.2.1).

Just as for products, the coproduct of two objects in a category $\mathcal{C}$ may not exist, or it may not be unique. The non-uniqueness is much less "bad" because given two candidate coproducts, they will be canonically isomorphic. They may not be equal, but they are isomorphic. But coproducts might not exist at all in certain categories. We will explore that a bit below.

Example 4.5.1.26. Consider the set $\mathbb{R}^2$ and partial order from Example 4.5.1.11 where $(x_1, y_1) \leqslant (x_2, y_2)$ if there exists $\ell \geqslant 1$ such that $x_1\ell = x_2$ and $y_1\ell = y_2$. Again the points $p := (1,0)$ and $q := (0,1)$ do not have a coproduct. Indeed, it would have to be a non-zero point that was on the same line-through-the origin as p and the same line-through-the-origin as q, of which there are none.

Exercise 4.5.1.27. Consider the preorder $\mathcal{P}$ of cards in a deck, shown in Example 3.4.1.3; it is not the entire story of cards in a deck, but take it to be so. In other words, be like a computer and take what's there at face value. Consider the preorder $\mathcal{P}$ as a category (by way of the functor **PrO** $\to$ **Cat**). For each of the following pairs, what is their coproduct in $\mathcal{P}$ (if it exists)?

a.) ⌜a diamond⌝$\sqcup$⌜a heart⌝ ? ⌜a queen⌝$\sqcup$⌜a black card⌝ ?

⌜a card⌝$\sqcup$⌜a red card⌝ ? ⌜a face card⌝$\sqcup$⌜a black card⌝ ?

b.) How would these answers differ if $\mathcal{P}$ was completed to the "whole story" partial order classifying cards in a deck?

◊

Exercise 4.5.1.28. Let X be a set, and consider it as a discrete category. Given two objects $x, y \in \mathrm{Ob}(X)$, under what conditions will there exist a coproduct $x \sqcup y$? ◊

Exercise 4.5.1.29. Consider the preorder $(\mathbb{N}, \texttt{divides})$, discussed in Exercise 4.5.1.4, where e.g. $5 \leqslant 15$ but $5 \not\leqslant 6$.

a.) What is the coproduct of 9 and 12 in that category?

b.) Is there a standard name for coproducts in that category?

◊

4.5.2 Diagrams in a category

We have been drawing diagrams since the beginning of the book. What is it that we have been drawing pictures *of*? The answer is that we have been drawing functors.

Definition 4.5.2.1. Let $\mathcal{C}$ and I be categories. [27] An *I-shaped diagram in $\mathcal{C}$* is simply a functor $d\colon I \to \mathcal{C}$. In this case I is called the *indexing category* for the diagram.

[27] In fact, the indexing category I is usually assumed to be small in the sense of Remark 4.1.1.2, meaning that its collection of objects is a set.

Suppose given an indexing category I and an I-shaped diagram $X\colon I \to \mathcal{C}$. One draws this as follows. For each object in $q \in I$, draw a dot labeled by $X(q)$; if several objects in I point to the same object in $\mathcal{C}$, then several dots will be labeled the same way. Draw the images of morphisms $f\colon q \to q'$ in I by drawing arrows between dots $X(q)$ and $X(q')$, and label each arrow by the image morphism $X(f)$ in $\mathcal{C}$. Again, if several morphisms in I are sent to the same morphism in $\mathcal{C}$, then several arrows will be labeled the same way. One can abbreviate this process by not drawing *every* morphism in I, so long as every morphism in I is represented by a unique path in $\mathcal{C}$, i.e. as long as the drawing is sufficiently unambiguous as a depiction of $X\colon I \to \mathcal{C}$.

Example 4.5.2.2. Consider the commutative diagram in **Set** drawn below:

$$\begin{tikzcd}
\mathbb{N} \arrow[r, "+1"] \arrow[d, "*2"'] & \mathbb{N} \arrow[d, "*2"] \\
\mathbb{N} \arrow[r, "+2"'] & \mathbb{Z}
\end{tikzcd} \tag{4.18}$$

This is the drawing of a functor $d\colon [1] \times [1] \to \mathbf{Set}$ (see Example 4.5.1.17). With notation for the objects and morphisms of $[1] \times [1]$ as shown in Diagram (4.17), we have $d(0,0) = d(0,1) = d(1,0) = \mathbb{N}$ and $d(1,1) = \mathbb{Z}$ (for some reason..) and $d(\mathrm{id}_0, f)\colon \mathbb{N} \to \mathbb{N}$ given by $n \mapsto n + 1$, etc.

The fact that d is a functor means it must respect composition formulas, which implies that Diagram (4.18) commutes. Recall from Section 2.2 that not all diagrams one can draw will commute; one must specify that a given diagram commutes if he or she wishes to communicate this fact. But then how is a *non-commuting diagram* to be understood as a functor?

Let $G \in \mathrm{Ob}(\mathbf{Grph})$ denote the following graph

$$\begin{tikzcd}
\overset{(0,0)}{\bullet} \arrow[r, "f"] \arrow[d, "h"'] & \overset{(0,1)}{\bullet} \arrow[d, "g"] \\
\overset{(1,0)}{\bullet} \arrow[r, "i"'] & \overset{(1,1)}{\bullet}
\end{tikzcd}$$

Recall the free category functor $F\colon \mathbf{Grph} \to \mathbf{Cat}$ from Example 4.1.2.30. The free category $F(G) \in \mathrm{Ob}(\mathbf{Cat})$ on G looks almost like $[1] \times [1]$ except that since $[f, g]$ is a different path in G than is $[h, i]$, they become different morphisms in $F(G)$. A functor $F(G) \to \mathbf{Set}$ might be drawn the same way that (4.18) is, but it would be a diagram that would *not* be said to commute.

We call $[1] \times [1]$ the *commutative square indexing category.* [28]

Exercise 4.5.2.3. Consider $[2]$, the linear order category of length 2.

a.) Is $[2]$ the appropriate indexing category for commutative triangles?

b.) If not, what is?

◊

[28] We might call what is here denoted by $F(G)$ the *noncommutative square indexing category.*

Example 4.5.2.4. Recall that an equalizer in **Set** was a diagram of sets that looked like this:

$$\begin{tikzcd} \overset{E}{\bullet} \arrow[r, "f"] & \overset{A}{\bullet} \arrow[r, shift left, "g_1"] \arrow[r, shift right, "g_2"'] & \overset{B}{\bullet} \end{tikzcd} \tag{4.19}$$

where $g_1 \circ f = g_2 \circ f$. What is the indexing category for such a diagram? It is the schema (4.19) with the PED $[f, g_1] \simeq [f, g_2]$. That is, in some sense you're seeing the indexing category, but the PED needs to be declared.

Exercise 4.5.2.5. Let $\mathcal{C}$ be a category, $A \in \text{Ob}(\mathcal{C})$ an object, and $f \colon A \to A$ a morphism in $\mathcal{C}$. Consider the two diagrams in $\mathcal{C}$ drawn below:

$$\boxed{\begin{tikzcd} \overset{A}{\bullet} \arrow[r, "f"] & \overset{A}{\bullet} \arrow[r, "f"] & \overset{A}{\bullet} \arrow[r, "f"] & \cdots \end{tikzcd}} \qquad \boxed{\begin{tikzcd} \overset{A}{\bullet} \arrow[loop left, "f"] \end{tikzcd}}$$

a.) Should these two diagrams have the same indexing category?

b.) If they should have the same indexing category, what is causing or allowing the pictures to appear different?

c.) If they should not have the same indexing category, what coincidence makes the two pictures have so much in common?

◊

Definition 4.5.2.6. Let $I \in \text{Ob}(\textbf{Cat})$ be a category. The *left cone on* I, denoted $I^{\triangleleft}$, is the category defined as follows. On objects we put $\text{Ob}(I^{\triangleleft}) = \{-\infty\} \sqcup \text{Ob}(I)$, and we call the new object $-\infty$ the *cone point of* $I^{\triangleleft}$. On morphisms we add a single new morphism $s_b \colon -\infty \to b$ for every object $b \in \text{Ob}(I)$; more precisely,

$$\text{Hom}_{I^{\triangleleft}}(a, b) = \begin{cases} \text{Hom}_I(a, b) & \text{if } a, b \in \text{Ob}(I) \\ \{s_b\} & \text{if } a = -\infty, b \in \text{Ob}(I) \\ \{\text{id}_{-\infty}\} & \text{if } a = b = -\infty \\ \emptyset & \text{if } a \in \text{Ob}(I), b = -\infty. \end{cases}$$

The composition formula is in some sense obvious. To compose two morphisms both in I, compose as dictated by I; if one has $-\infty$ as source then there will be a unique choice of composite.

There is an obvious inclusion of categories,

$$I \to I^{\triangleleft}. \tag{4.20}$$

Remark 4.5.2.7. Note that the specification of $I^{\triangleleft}$ given in Definition 4.5.2.6 works just as well if I is considered a schema and we are constructing a schema $I^{\triangleleft}$: add the new object $-\infty$ and the new arrows $s_b \colon -\infty \to b$ for each $b \in \text{Ob}(I)$, and for every morphism $f \colon b \to b'$ in I add a PED $[s_{b'}] \simeq [s_b, f]$. We generally will not distinguish between categories and schemas, since they are equivalent.

Example 4.5.2.8. For a natural number $n \in \mathbb{N}$, we define the *n-leaf star schema*, denoted $\textbf{Star}_n$, to be the category (or schema, see Remark 4.5.2.7) $\underline{n}^{\triangleleft}$, where $\underline{n}$ is the discrete category on n objects. Below we draw $\mathbf{Star}_0, \mathbf{Star}_1, \mathbf{Star}_2$, and $\mathbf{Star}_3$.

$\mathbf{Star}_0$

$$\begin{tikzcd}\overset{-\infty}{\bullet}\end{tikzcd}$$

$\mathbf{Star}_1$

$$\begin{tikzcd}\overset{-\infty}{\bullet} \arrow[d, "s_1"'] \\ \overset{1}{\bullet}\end{tikzcd}$$

$\mathbf{Star}_2$

$$\begin{tikzcd} & \overset{-\infty}{\bullet} \arrow[dl, "s_1"'] \arrow[dr, "s_2"] & \\ \overset{1}{\bullet} & & \overset{2}{\bullet}\end{tikzcd}$$

$\mathbf{Star}_3$

$$\begin{tikzcd} & \overset{-\infty}{\bullet} \arrow[dl, "s_1"'] \arrow[d, "s_2"] \arrow[dr, "s_3"] & \\ \overset{1}{\bullet} & \overset{2}{\bullet} & \overset{3}{\bullet}\end{tikzcd}$$

Exercise 4.5.2.9. Let $\mathcal{C}_0 := \underline{0}$ denote the empty category and for any natural number $n \in \mathbb{N}$, let $\mathcal{C}_{n+1} = (\mathcal{C}_n)^{\triangleleft}$. Draw $\mathcal{C}_4$. ◊

Exercise 4.5.2.10. Let $\mathcal{C}$ be the graph indexing schema as in (4.7). What is $\mathcal{C}^{\triangleleft}$ and how does it compare to (4.19)? ◊

Definition 4.5.2.11. Let $I \in \mathrm{Ob}(\mathbf{Cat})$ be a category. The *right cone on* I, denoted $I^{\triangleright}$, is the category defined as follows. On objects we put $\mathrm{Ob}(I^{\triangleright}) = \mathrm{Ob}(I) \sqcup \{\infty\}$, and we call the new object ∞ the *cone point of* $I^{\triangleright}$. On morphisms we add a single new morphism $t_b \colon b \to \infty$ for every object $b \in \mathrm{Ob}(I)$; more precisely,

$$\mathrm{Hom}_{I^{\triangleright}}(a,b) = \begin{cases} \mathrm{Hom}_I(a,b) & \text{if } a,b \in \mathrm{Ob}(I) \\ \{t_b\} & \text{if } a \in \mathrm{Ob}(I), b = \infty \\ \{\mathrm{id}_\infty\} & \text{if } a = b = \infty \\ \varnothing & \text{if } a = \infty, b \in \mathrm{Ob}(I). \end{cases}$$

The composition formula is in some sense obvious. To compose two morphisms both in I, compose as dictated by I; if one has ∞ as target then there will be a unique choice of composite.

There is an obvious inclusion of categories $I \to I^{\triangleright}$.

Exercise 4.5.2.12. Let $\mathcal{C}$ be the category $(\underline{2}^{\triangleleft})^{\triangleright}$, where $\underline{2}$ is the discrete category on two objects. Then $\mathcal{C}$ is somehow square-shaped, but what category is it exactly? Looking at Example 4.5.2.2, is $\mathcal{C}$ the commutative diagram indexing category $[1] \times [1]$, is it the non-commutative diagram indexing category $F(G)$, or is it something else? ◊

4.5.3 Limits and colimits in a category

Let $\mathcal{C}$ be a category, let I be an indexing category (which just means that I is a category that we're about to use as the indexing category for a diagram), and let $D \colon I \to \mathcal{C}$ an I-shaped diagram (which just means a functor). It is in relation to this setup that we can discuss the limit or colimit. In general the limit of a diagram $D \colon I \to \mathcal{C}$ will be a $I^{\triangleleft}$ shaped diagram $\lim D \colon I^{\triangleleft} \to \mathcal{C}$. In the case of products $I = \underline{2}$ and $I^{\triangleleft} = \mathbf{Star}_2$ looks like a span (see Example 4.5.2.8). But out of all the $I^{\triangleleft}$-shaped diagrams, which is the limit of D? Answer: the one with the universal "gateway" property, see Remark 4.5.1.9.

4.5.3.1 Universal objects

Definition 4.5.3.2. Let $\mathcal{C}$ be a category. An object $a \in \mathrm{Ob}(\mathcal{C})$ is called *initial* if, for all objects $c \in \mathrm{Ob}(\mathcal{C})$ there exists a unique morphism $a \to c$, i.e. $|\mathrm{Hom}_{\mathcal{C}}(a,c)| = 1$. An object $z \in \mathrm{Ob}(\mathcal{C})$ is called *terminal* if, for all objects $c \in \mathrm{Ob}(\mathcal{C})$ there is exists a unique morphism $c \to z$, i.e. $|\mathrm{Hom}_{\mathcal{C}}(c,z)| = 1$.

An object in a category is called *universal* if it is either initial or terminal, but we rarely use that term in practice, preferring to be specific about whether the object is initial or terminal. The word *final* is synonymous with the word terminal, but we'll try to constantly use terminal.

Colimits will end up being defined as initial things of a certain sort, and limits will end up being defined as terminal things of a certain sort. But we will get to that in Section 4.5.3.15.

Warning 4.5.3.3. A category $\mathcal{C}$ may have more than one initial object; similarly a category $\mathcal{C}$ may have more than one terminal object. We will see in Example 4.5.3.5 that any set with one element, e.g. $\{*\}$ or $\{☺\}$, is a terminal object in **Set**. These terminal sets have the same number of elements, but they are not the exact-same set; two sets having the same cardinality means precisely that there exists an isomorphism between them.

In fact, Proposition 4.5.3.4 below shows that in any category $\mathcal{C}$, any two terminal objects in $\mathcal{C}$ are isomorphic (similarly, any two initial objects in $\mathcal{C}$ are isomorphic). While there are many isomorphisms in **Set** between $\{1, 2, 3\}$ and $\{a, b, c\}$, there is only one isomorphism between $\{*\}$ and ☺. This is always the case for universal objects: there is a unique isomorphism between any two terminal (respectively initial) objects in any category.

As a result, people often speak of *the* initial object in $\mathcal{C}$ or *the* terminal object in $\mathcal{C}$, as though there was only one. "It's unique up to unique ismorphism!" is the justification for this use of the so-called definite article *the* rather than the indefinite article *a*. This is not a very misleading way of speaking, because just like the president today does not contain exactly the same atoms as the president yesterday, the difference is unimportant. But we still mention this as a warning: if $\mathcal{C}$ has a terminal object, we may speak of it as though it were unique, calling it *the terminal object*, and similarly for initial objects.

We will use the definite article throughout this document, e.g. in Example 4.5.3.5 we will discuss the initial object in **Set** and the terminal object in **Set**. This is common throughout mathematical literature as well.

Proposition 4.5.3.4. *Let $\mathcal{C}$ be a category and let $a_1, a_2 \in \mathrm{Ob}(\mathcal{C})$ both be initial objects. Then there is a unique isomorphism $a_1 \xrightarrow{\cong} a_2$. (Similarly, for any two terminal objects in $\mathcal{C}$ there is a unique isomorphism between them.)*

Proof. Suppose a_1 and a_2 are initial. Since a_1 is initial there is a unique morphism $f\colon a_1 \to a_2$; there is also a unique morphism $a_1 \to a_1$, which must be id_{a_1}. Since a_2 is initial there is a unique morphism $g\colon a_2 \to a_1$; there is also a unique morphism $a_2 \to a_2$, which must be id_{a_2}. So $g \circ f = \mathrm{id}_{a_1}$ and $f \circ g = \mathrm{id}_{a_2}$, which means that f is the desired (unique) isomorphism.

The proof for terminal objects is appropriately "dual".

□

Example 4.5.3.5. The initial object in **Set** is the set a for which there is always one way to map from a to anything else. Given $c \in \mathrm{Ob}(\mathbf{Set})$ there is exactly one function $\varnothing \to c$, because there are no choices to be made, so the empty set $\varnothing$ is the initial object in **Set**.

The terminal object in **Set** is the set z for which there is always one way to map to z from anything else. Given $c \in \mathrm{Ob}(\mathbf{Set})$ there is exactly one function $c \to \{☺\}$, where $\{☺\}$ is any set with one element, because there are no choices to be made: everything in c must be sent to the single element in $\{☺\}$. There are lots of terminal objects in **Set**, and they are all isomorphic to $\underline{1}$.

Example 4.5.3.6. The initial object in **Grph** is the graph a for which there is always one way to map from a to anything else. Given $c \in \mathrm{Ob}(\mathbf{Grph})$, there is exactly one function $\emptyset \to c$, where $\emptyset \in \mathbf{Grph}$ is the empty graph; so $\emptyset$ is the initial object.

The terminal object in **Grph** is more interesting. It is $\mathcal{L}oop$, the graph with one vertex and one arrow. In fact there are infinitely many terminal objects in **Grph**, but all of them are isomorphic to $\mathcal{L}oop$.

Exercise 4.5.3.7. Let X be a set, let $\mathbb{P}(X)$ be the set of subsets of X (see Definition 2.7.4.9). We can regard $\mathbb{P}(X)$ as a preorder under inclusion of subsets (see for example Section 3.4.2). And we can regard preorders as categories using a functor $\mathbf{PrO} \to \mathbf{Cat}$ (see Proposition 4.2.1.17).

a.) What is the initial object in $\mathbb{P}(X)$?

b.) What is the terminal object in $\mathbb{P}(X)$?

◊

Example 4.5.3.8. The initial object in the category **Mon** of monoids is the trivial monoid, $\underline{1}$. For any monoid M, a morphism of monoids $\underline{1} \to M$ is a functor between 1-object categories and these are determined by where they send morphisms. Since $\underline{1}$ has only the identity morphism and functors must preserve identities, there is no choice involved in finding a monoid morphism $\underline{1} \to M$.

Similarly, the terminal object in **Mon** is also the trivial monoid, $\underline{1}$. For any monoid M, a morphism of monoids $M \to \underline{1}$ sends everything to the identity; there is no choice.

Exercise 4.5.3.9.

a.) What is the initial object in **Grp**, the category of groups?

b.) What is the terminal object in **Grp**?

◊

Example 4.5.3.10. Recall the preorder **Prop** of logical propositions from Section 4.2.4.1. The initial object is a proposition that implies all others. It turns out that "FALSE" is such a proposition. The proposition "FALSE" is like "$1 \neq 1$"; in logical formalism it can be shown that if "FALSE" is true then everything is true.

The terminal object in **Prop** is a proposition that is implied by all others. It turns out that "TRUE" is such a proposition. In logical formalism, everything implies that "TRUE" is true.

Example 4.5.3.11. The discrete category $\underline{2}$ has no initial object and no terminal object. The reason is that it has two objects $1, 2$, but no maps from one to the other, so $\mathrm{Hom}_{\underline{2}}(1,2) = \mathrm{Hom}_{\underline{2}}(2,1) = \emptyset$.

Exercise 4.5.3.12. Recall the `divides` preorder from Exercise 4.5.1.4, where 5 `divides` 15.

a.) Considering this preorder as a category, does it have an initial object?

b.) Does it have a terminal object?

◊

Exercise 4.5.3.13. Let $\mathcal{M} = (\mathrm{List}(\{a,b\}), [\,], +\!\!+)$ denote the free monoid on $\{a,b\}$ (see Definition 3.1.1.15), considered as a category (via Theorem 4.2.1.3).

a.) Does it have an initial object?

b.) Does it have a terminal object?

c.) Which monoids have initial (respectively terminal) objects?

◊

Exercise 4.5.3.14. Let S be a set and consider the indiscrete category $K_S \in \mathrm{Ob}(\mathbf{Cat})$ on objects S (see Example 4.3.4.3).

a.) For what S does K_S have an initial object?

b.) For what S does K_S have a terminal object?

◊

4.5.3.15 Examples of limits

Let $\mathcal{C}$ be a category and let $X, Y \in \mathrm{Ob}(\mathcal{C})$ be objects. Definition 4.5.1.8 defines a product of X and Y to be a span $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$ such that for every other span $X \xleftarrow{p} Z \xrightarrow{q} Y$ there exists a unique morphism $Z \to X \times Y$ making the triangles commute. It turns out that we can enunciate this in our newly formed language of universal objects by saying that the span $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$ is itself a terminal object in the category of spans on X and Y. Phrasing the definition of products in this way will be generalizable to defining arbitrary limits.

Construction 4.5.3.16 (Products). Let $\mathcal{C}$ be a category and let X_1, X_2 be objects. We can consider this setup as a diagram $X\colon \underline{2} \to \mathcal{C}$, where $X(1) = X_1$ and $X(2) = X_2$. Consider the category $\underline{2}^{\triangleleft} = \mathbf{Star}_2$, which is drawn in Example 4.5.2.8; the inclusion $i\colon \underline{2} \to \underline{2}^{\triangleleft}$, as in (4.20); and the category of functors $\mathrm{Fun}(\underline{2}^{\triangleleft}, \mathcal{C})$. The objects in $\mathrm{Fun}(\underline{2}^{\triangleleft}, \mathcal{C})$ are spans in $\mathcal{C}$ and the morphisms are natural transformations between them. Given a functor $S\colon \underline{2}^{\triangleleft} \to \mathcal{C}$ we can compose with $i\colon \underline{2} \to \underline{2}^{\triangleleft}$ to get a functor $\underline{2} \to \mathcal{C}$. We want that to be X.

$$\begin{tikzcd}
\underline{2} \arrow[r, "X"] \arrow[d, "i"'] & \mathcal{C} \\
\underline{2}^{\triangleleft} \arrow[ur, "S"'] &
\end{tikzcd}$$

So we are ready to define the category of spans on X_1 and X_2.

Define the *category of spans on* X, denoted $\mathcal{C}_{/X}$, to be the category whose objects and morphisms are as follows:

$$\begin{aligned}
\mathrm{Ob}(\mathcal{C}_{/X}) &= \{S\colon \underline{2}^{\triangleleft} \to \mathcal{C} \mid S \circ i = X\} \\
\mathrm{Hom}_{\mathcal{C}_{/X}}(S, S') &= \{\alpha\colon S \to S' \mid \alpha \circ i = \mathrm{id}_X\}.
\end{aligned} \tag{4.21}$$

The product of X_1 and X_2 was defined in Definition 4.5.1.8; we can now recast $X_1 \times X_2$ as the terminal object in $\mathcal{C}_{/X}$.

To bring this down to earth, an object in $\mathcal{C}_{/X}$ can be pictured as a diagram in $\mathcal{C}$ of the following form:

$$\begin{tikzcd}
& Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\
X_1 & & X_2
\end{tikzcd}$$

In other words, the objects of $\mathcal{C}_{/X}$ are spans, each of which we might write in-line as $X_1 \xleftarrow{p} Z \xrightarrow{q} X_2$. A morphism in $\mathcal{C}_{/X}$ from object $X_1 \xleftarrow{p} Z \xrightarrow{q} X_2$ to object $X_1 \xleftarrow{p'} Z' \xrightarrow{q'} X_2$ consists of a morphism $\ell \colon Z \to Z'$, such that $p' \circ \ell = p$ and $q' \circ \ell = q$. So the set of such morphisms in $\mathcal{C}_{/X}$ are all the ℓ's that make the right-hand diagram commute: [29]

$$\mathrm{Hom}_{\mathcal{C}_{/X}}\left(\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\ X_1 & & X_2 \end{tikzcd}, \begin{tikzcd} & Z' \arrow[dl, "p'"'] \arrow[dr, "q'"] & \\ X_1 & & X_2 \end{tikzcd}\right) = \left\{\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] \arrow[dd, dashed, "\ell"] & \\ X_1 & & X_2 \\ & Z' \arrow[ul, "p'"] \arrow[ur, "q'"'] & \end{tikzcd}\right\} \tag{4.22}$$

Each object in $\mathcal{C}_{/X}$ is a span on X_1 and X_2, and each morphism in $\mathcal{C}_{/X}$ is a "morphism of cone points in $\mathcal{C}$ making everything in sight commute". The terminal object in $\mathcal{C}_{/X}$ is the product of X_1 and X_2; see Definition 4.5.1.8.

It may be strange to have a category in which the objects are spans in another category. But once you admit this possibility, the notion of morphism between spans is totally sensible. Or if it isn't, then stare at (4.22) for 30 seconds and say to yourself "When in Rome..!" These are the aqueducts of category theory, and they work wonders.

Example 4.5.3.17. Consider the arbitrary 6-object category $\mathcal{C}$ drawn below, in which the

[29] To be completely pedantic, according to (4.21), the morphisms in $\mathcal{C}_{/X}$ should be drawn like this:

$$\mathrm{Hom}_{\mathcal{C}_{/X}}\left(\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\ X_1 & & X_2 \end{tikzcd}, \begin{tikzcd} & Z' \arrow[dl, "p'"'] \arrow[dr, "q'"] & \\ X_1 & & X_2 \end{tikzcd}\right) = \left\{\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] \arrow[ddd, dashed, "\alpha_{-\infty}"] & \\ X_1 \arrow[d, equal, "\alpha_1"'] & & X_2 \arrow[d, equal, "\alpha_2"] \\ X_1 & & X_2 \\ & Z' \arrow[ul, "p'"] \arrow[ur, "q'"'] & \end{tikzcd}\right\}$$

But this is going a bit overboard. The point is, the set $\mathrm{Hom}_{\mathcal{C}_{/X}}$ is the set of morphisms serving the role of $\alpha_{-\infty} \colon Z \to Z'$.

three diagrams that can commute do:

$$\mathcal{C} := \boxed{\begin{tikzcd} & & X_1 & & \\ A \arrow[urr, bend left, "a"] \arrow[ur, phantom, "\checkmark"] & B \arrow[l, "f"'] \arrow[ur, "b_1"] \arrow[dr, "b_2"] & & C \arrow[ul, "c_1"'] \arrow[dl, "c_2"'] \arrow[r, "g"] \arrow[ur, phantom, "\checkmark"] \arrow[dr, phantom, "\checkmark"] & D \arrow[ull, bend right, "d_1"'] \arrow[dll, bend left, "d_2"] \\ & & X_2 & & \end{tikzcd}}$$

Let $X\colon \underline{2} \to \mathcal{C}$ be given by $X(1) = X_1$ and $X(2) = X_2$. Then the category of spans on X might be drawn

$$\mathcal{C}_{/X} \cong \boxed{\begin{tikzcd} (B,b_1,b_2) & (C,c_1,c_2) \arrow[r, "g"] & (D,d_1,d_2) \end{tikzcd}}$$

4.5.3.18 Definition of limit

Definition 4.5.3.19. Let $\mathcal{C}$ be a category, let I be a category; let $I^{\lhd}$ be the left cone on I, and let $i\colon I \to I^{\lhd}$ be the inclusion. Suppose that $X\colon I \to \mathcal{C}$ is an I-shaped diagram in $\mathcal{C}$. The *slice category of $\mathcal{C}$ over X* denoted $\mathcal{C}_{/X}$ is the category whose objects and morphisms are as follows:

$$\mathrm{Ob}(\mathcal{C}_{/X}) = \{S\colon I^{\lhd} \to \mathcal{C} \mid S \circ i = X\}$$
$$\mathrm{Hom}_{\mathcal{C}_{/X}}(S, S') = \{\alpha\colon S \to S' \mid \alpha \circ i = \mathrm{id}_X\}.$$

A *limit of X*, denoted $\lim_I X$ or $\lim X$, is a terminal object in $\mathcal{C}_{/X}$.

Pullbacks The relevant indexing category for pullbacks is the cospan, $I = \underline{2}^{\rhd}$ drawn as to the left below:

$$I \quad \boxed{\begin{tikzcd} 0 \arrow[dr] & & 1 \arrow[dl] \\ & 2 & \end{tikzcd}} \qquad X\colon I \to \mathcal{C} \quad \begin{tikzcd} X_0 \arrow[dr] & & X_1 \arrow[dl] \\ & X_2 & \end{tikzcd}$$ [30]

A I-shaped diagram in $\mathcal{C}$ is a functor $X\colon I \to \mathcal{C}$, which we might draw as to the right above (e.g. $X_0 \in \mathrm{Ob}(\mathcal{C})$).

[30]We use a dash box here because we're not drawing the whole category but merely a diagram existing inside $\mathcal{C}$.

An object S in the slice category $\mathcal{C}_{/X}$ is a commutative diagram $S\colon I^{\triangleleft} \to \mathcal{C}$ over X, which looks like the box to the left below:

$S \in \mathrm{Ob}(\mathcal{C}_{/X})$

$$\begin{tikzcd}
 & S_{-\infty} \arrow[dl] \arrow[dr] & \\
\overset{X_0}{\bullet} \arrow[dr] & & \overset{X_1}{\bullet} \arrow[dl] \\
 & \overset{X_2}{\bullet} &
\end{tikzcd}$$

$f\colon S \to S'$

$$\begin{tikzcd}
 & S_{-\infty} \arrow[d, "f"] \arrow[ddl, bend right] \arrow[ddr, bend left] & \\
 & S'_{-\infty} \arrow[dl] \arrow[dr] & \\
\overset{X_0}{\bullet} \arrow[dr] & & \overset{X_1}{\bullet} \arrow[dl] \\
 & \overset{X_2}{\bullet} &
\end{tikzcd}$$

A morphism in $\mathcal{C}_{/X}$ is drawn in the dashbox to the right above. A terminal object in $\mathcal{C}_{/X}$ is precisely the "gateway" we want, i.e. the limit of X is the pullback $X_0 \times_{X_2} X_1$.

Exercise 4.5.3.20. Let I be the graph indexing category (see 4.7).

a.) What is $I^{\triangleleft}$?

b.) Now let $G\colon I \to \mathbf{Set}$ be the graph from Example 3.3.1.2. Give an example of an object in $\mathbf{Set}_{/G}$.

c.) We have already given a name to the limit of $G\colon I \to \mathbf{Set}$; what is it?

$\Diamond$

Exercise 4.5.3.21. Let $\mathcal{C}$ be a category and let $I = \varnothing$ be the empty category. There is a unique functor $X\colon \varnothing \to \mathcal{C}$.

a.) What is the slice category $\mathcal{C}_{/X}$?

b.) What is the limit of X?

$\Diamond$

Example 4.5.3.22. Often one wants to take the limit of some strange diagram. We have now constructed the limit for any shape diagram. For example, if we want to take the product of more than two, say n, objects, we could use the diagram shape $I = \underline{n}$ whose cone is $\mathbf{Star}_n$ from Example 4.5.2.8.

Example 4.5.3.23. We have now defined limits in any category, so we have defined limits in $\mathbf{Cat}$. Let $[1]$ denote the category depicted

$$\begin{tikzcd}
\overset{0}{\bullet} \arrow[r, "e"] & \overset{1}{\bullet}
\end{tikzcd}$$

and let $\mathcal{C}$ be a category. Naming two categories is the same thing as naming a functor $X\colon \underline{2} \to \mathbf{Cat}$, so we now have such a functor. Its limit is denoted $[1] \times \mathcal{C}$. It turns out that $[1] \times \mathcal{C}$ looks like a "$\mathcal{C}$-shaped prism". It consists of two panes, front and back say, each having the precise shape as $\mathcal{C}$ (same objects, same arrows, same composition), and morphisms from the front pane to the back pane making all front-to-back squares commute. For example, if $\mathcal{C}$ looked was the category generated by the schema to the left below, then $\mathcal{C} \times [1]$ would be the category generated by the schema to the right below:

$$\begin{tikzcd}
\overset{A}{\bullet} \arrow[r, "f"] \arrow[d, "g"'] & \overset{B}{\bullet} \arrow[d, "h"] \\
\overset{C}{\bullet} & \overset{D}{\bullet}
\end{tikzcd}
\qquad
\begin{tikzcd}
& \overset{A1}{\bullet} \arrow[rr, "f1"] \arrow[dd, "g1"'] & & \overset{B1}{\bullet} \arrow[dd, "h1"] \\
\overset{A0}{\bullet} \arrow[ur, "Ae"] \arrow[rr, "f0"] \arrow[dd, "g0"'] & & \overset{B0}{\bullet} \arrow[ur, "Be"] \arrow[dd, "h0"] & \\
& \overset{C1}{\bullet} & & \overset{D1}{\bullet} \\
\overset{C0}{\bullet} \arrow[ur, "Ce"'] & & \overset{D0}{\bullet} \arrow[ur, "De"'] &
\end{tikzcd}$$

It turns out that a natural transformation $\alpha \colon F \to G$ between functors $F, G \colon \mathcal{C} \to \mathcal{D}$ is the same thing as a functor $\mathcal{C} \times [1] \to \mathcal{D}$ such that the front pane is sent via F and the back pane is sent via G. The components are captured by the front-to-back morphisms, and the naturality is captured by the commutativity of the front-to-back squares in $\mathcal{C} \times [1]$.

Remark 4.5.3.24. Recall in Section 2.7.6.6 we described relative sets. In fact, Definition 2.7.6.7 basically defines a category of relative sets over any fixed set B. Let $\underline{1}$ denote the discrete category on one object, and note that providing a functor $\underline{1} \to \mathbf{Set}$ is the same as simply providing a set, so consider $B \colon \underline{1} \to \mathbf{Set}$. Then the slice category $\mathbf{Set}_{/B}$, as defined in Definition 4.5.3.19 is precisely the category of relative sets over B: it has the same objects and morphisms as was described in Definition 2.7.6.7.

4.5.3.25 Definition of colimit

The definition of colimits is appropriately "dual" to the definition of limits. Instead of looking at left cones, we look at right cones; instead of being interested in terminal objects, we are interested in initial objects.

Definition 4.5.3.26. Let $\mathcal{C}$ be a category, let I be a category; let $I^{\rhd}$ be the right cone on I, and let $i \colon I \to I^{\rhd}$ be the inclusion. Suppose that $X \colon I \to \mathcal{C}$ is an I-shaped diagram in $\mathcal{C}$. The *coslice category of $\mathcal{C}$ over X* denoted $\mathcal{C}_{X/}$ is the category whose objects and morphisms are as follows:

$$\mathrm{Ob}(\mathcal{C}_{X/}) = \{S \colon I^{\rhd} \to \mathcal{C} \mid S \circ i = X\}$$
$$\mathrm{Hom}_{\mathcal{C}_{X/}}(S, S') = \{\alpha \colon S \to S' \mid \alpha \circ i = \mathrm{id}_X\}.$$

A *colimit of X*, denoted $\mathrm{colim}_I X$ or $\mathrm{colim}\, X$, is an initial object in $\mathcal{C}_{X/}$.

Pushouts The relevant indexing category for pushouts is the span, $I = \underline{2}^{\lhd}$ drawn as to the left below:

I

$$\begin{tikzcd}
1 & & 2 \\
& 0 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

$X\colon I \to \mathcal{C}$

$$\begin{tikzcd}
X_1 & & X_2 \\
& X_0 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

An I-shaped diagram in $\mathcal{C}$ is a functor $X\colon I \to \mathcal{C}$, which we might draw as to the right above (e.g. $X_0 \in \mathrm{Ob}(\mathcal{C})$).

An object S in the coslice category $\mathcal{C}_{X/}$ is a commutative diagram $S\colon I^{\rhd} \to \mathcal{C}$ over X, which looks like the box to the left below:

$S \in \mathrm{Ob}(\mathcal{C}_{X/})$

$$\begin{tikzcd}
& S_\infty & \\
X_1 \arrow[ur] & & X_2 \arrow[ul] \\
& X_0 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

$f\colon S \to S'$

$$\begin{tikzcd}
& S'_\infty & \\
& S_\infty \arrow[u, "f"'] & \\
X_1 \arrow[ur] \arrow[uur, bend left] & & X_2 \arrow[ul] \arrow[uul, bend right] \\
& X_0 \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

A morphism in $\mathcal{C}_{X/}$ is drawn in the dashbox to the right above. An initial object in $\mathcal{C}_{X/}$ is precisely the "gateway" we want; i.e. the colimit of X is the pushout, $X_1 \sqcup_{X_0} X_2$.

Exercise 4.5.3.27. Let I be the graph indexing category (see 4.7).

a.) What is $I^{\rhd}$?

b.) Now let $G\colon I \to \mathbf{Set}$ be the graph from Example 3.3.1.2. Give an example of an object in $\mathbf{Set}_{G/}$.

c.) We have already given a name to the colimit of $G\colon I \to \mathbf{Set}$; what is it?

◊

Exercise 4.5.3.28. Let $\mathcal{C}$ be a category and let $I = \varnothing$ be the empty category. There is a unique functor $X\colon \varnothing \to \mathcal{C}$.

a.) What is the coslice category $\mathcal{C}_{X/}$?

b.) What is the colimit of X (assuming it exists)?

◊

Example 4.5.3.29 (Cone as colimit). We have now defined colimits in any category, so we have defined colimits in **Cat**. Let $\mathcal{C}$ be a category and recall from Example 4.5.3.23 the category $\mathcal{C} \times [1]$. The inclusion of the front pane is a functor $i_0 \colon \mathcal{C} \to \mathcal{C} \times [1]$ (similarly, the inclusion of the back pane is a functor $i_1 \colon \mathcal{C} \to \mathcal{C} \times [1]$). Finally let $t \colon \mathcal{C} \to \underline{1}$ be the unique functor to the terminal category (see Exercise 4.1.2.37). We now have a diagram in **Cat** of the form

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "i_0"] \arrow[d, "t"'] & \mathcal{C} \times [1] \\
\underline{1} &
\end{tikzcd}$$

The colimit (i.e. the pushout) of this diagram in **Cat** slurps down the entire front pane of $\mathcal{C} \times [1]$ to a point, and the resulting category is isomorphic to $\mathcal{C}^{\triangleleft}$. Figure 4.23 is a drawing of this phenomenon.

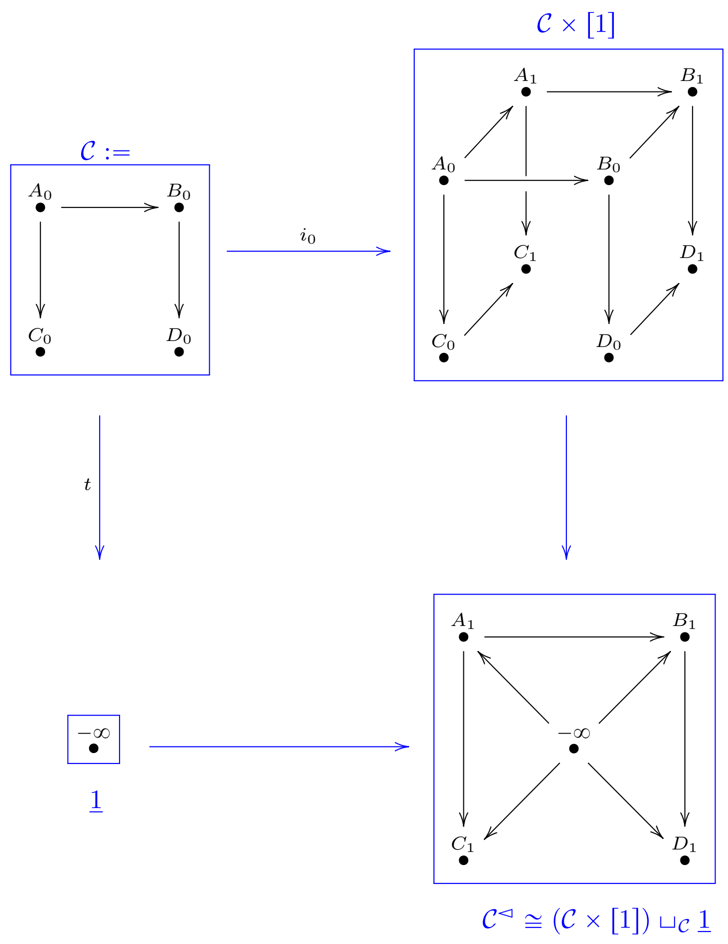
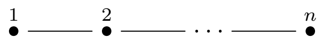


Figure 4.23: Let $\mathcal{C}$ be the category drawn in the upper left corner. The left cone $\mathcal{C}^{\triangleleft}$ on $\mathcal{C}$ is obtained as a pushout in **Cat**. We first make a prism $\mathcal{C} \times [1]$, and then identify the front pane with a point.

(Similarly, the pushout of the analogous diagram for i_1 would give $\mathcal{C}^{\triangleright}$.)

Example 4.5.3.30. Consider the category **Top** of topological spaces. The (hollow) circle is a topological space which people often denote S^1 (for "1-dimensional sphere"). The filled-in circle, also called a 2-dimensional disk, is denoted D^2. The inclusion of the circle into the disk is continuous so we have a morphism in **Top** of the form $i\colon S^1 \to D^2$. The terminal object in **Top** is the one-point space $\{☺\}$, and so there is a unique morphism $t\colon S^1 \to \{☺\}$. The pushout of the diagram $D^2 \xleftarrow{i} S^1 \xrightarrow{t} \{☺\}$ is isomorphic to the 2-dimensional sphere (the exterior of a tennis ball), S^2. The reason is that we have slurped the entire bounding circle to a point, and the category of topological spaces has the right morphisms to ensure that the resulting space really is a sphere.

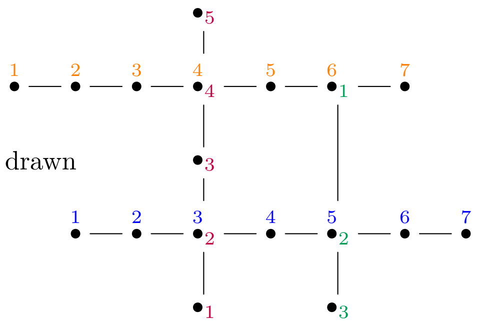
Application 4.5.3.31. Consider the symmetric graph G_n consisting of a chain of n vertices,



Think of this as modeling a subway line. There are n-many graph homomorphisms $G_1 \to G_n$ given by the various vertices. One can create transit maps using colimits. For example, the colimit of the diagram to the left is the symmetric graph drawn to the right below.

$$\mathrm{colim}\left(\begin{tikzcd} G_1 \arrow[r, "4"] \arrow[d, "4"'] & G_7 & G_1 \arrow[l, "6"'] \arrow[d, "1"] \\ G_5 & & G_3 \\ G_1 \arrow[u, "2"] \arrow[r, "3"'] & G_7 & G_1 \arrow[l, "5"] \arrow[u, "2"'] \end{tikzcd}\right)$$

can be drawn



◊◊

4.6 Other notions in Cat

In this section we discuss some leftover notions about categories. For example in Section 4.6.1 we explain a kind of duality for categories, in which arrows are flipped. For example reversing the order in a preorder is an example of this duality, as is the similarity between limits and colimits. In Section 4.6.2 we discuss the so-called Grothendieck construction which in some sense graphs functors, and we show that it is useful for transforming databases into the kind of format (RDF) used in scraping data off webpages. We define a general construction for creating categories in Section 4.6.4. Finally, in Section 4.6.5 we show that precisely the same arithmetic statements that held for sets in Section 2.7.3 hold for categories.

4.6.1 Opposite categories

People used to discuss two different kinds of functors between categories: the so-called *covariant functors* and the so-called *contravariant functors*. Covariant functors are what we have been calling functors. The reader may have come across the idea of contravariance when considering Exercise 4.2.3.2.[31] There we saw that a continuous mapping of topological spaces $f\colon X \to Y$ does not induce a morphism of orders on their open sets $\mathrm{Open}(X) \to \mathrm{Open}(Y)$; that is not required by the notion of continuity. Instead, a morphism of topological spaces $f\colon X \to Y$ induces a morphism of orders $\mathrm{Open}(Y) \to \mathrm{Open}(X)$, going backwards. So we do not have a functor $\mathbf{Top} \to \mathbf{PrO}$ in this way, but it's quite close. One used to say that Open is a *contravariant functor* $\mathbf{Top} \to \mathbf{PrO}$.

[31] Similarly, see Exercise 4.2.4.4.

As important and common as contravariance is, people found that keeping track of which functors were covariant and which were contravariant was a big hassle. Luckily, there is a simple work-around, which simplifies everything: the notion of opposite categories.

Definition 4.6.1.1. Let $\mathcal{C}$ be a category. The *opposite category* of $\mathcal{C}$, denoted $\mathcal{C}^{\text{op}}$, has the same objects as $\mathcal{C}$, i.e. $\text{Ob}(\mathcal{C}^{\text{op}}) = \text{Ob}(\mathcal{C})$, and for any two objects c, c', one defines

$$\text{Hom}_{\mathcal{C}^{\text{op}}}(c, c') := \text{Hom}_{\mathcal{C}}(c', c).$$

Example 4.6.1.2. If $n \in \mathbb{N}$ is a natural number and $\underline{n}$ the corresponding discrete category, then $\underline{n}^{\text{op}} = \underline{n}$. Recall the span category $I = \underline{2}^{\triangleleft}$ from Definition 4.5.1.8. Its opposite is the cospan category $I^{\text{op}} = \underline{2}^{\triangleright}$, from Definition 4.5.1.23.

Exercise 4.6.1.3. Let $\mathcal{C}$ be the category from Example 4.5.3.17. Draw $\mathcal{C}^{\text{op}}$. ◊

Lemma 4.6.1.4. *Let $\mathcal{C}$ and $\mathcal{D}$ be categories. One has $(\mathcal{C}^{\text{op}})^{\text{op}} = \mathcal{C}$. Also we have $\text{Fun}(\mathcal{C}, \mathcal{D}) \cong \text{Fun}(\mathcal{C}^{\text{op}}, \mathcal{D}^{\text{op}})$. This implies that a functor $\mathcal{C}^{\text{op}} \to \mathcal{D}$ can be identified with a functor $\mathcal{C} \to \mathcal{D}^{\text{op}}$.*

Proof. This follows straightforwardly from the definitions. □

Exercise 4.6.1.5. In Exercises 4.2.3.2, 4.2.4.3, and 4.2.4.4 there were questions about whether a certain function $\text{Ob}(\mathcal{C}) \to \text{Ob}(\mathcal{D})$ extended to a functor $\mathcal{C} \to \mathcal{D}$. In each case, see if the proposed function would extend to a "contravariant functor" i.e. to a functor $\mathcal{C}^{\text{op}} \to \mathcal{D}$. ◊

Example 4.6.1.6 (Simplicial sets). Recall from Example 4.3.4.4 the category $\boldsymbol{\Delta}$ of linear orders $[n]$. For example, $[1]$ is the linear order $0 \leqslant 1$ and $[2]$ is the linear order $0 \leqslant 1 \leqslant 2$. Both $[1]$ and $[2]$ are objects of $\boldsymbol{\Delta}$. There are 6 morphisms from $[1]$ to $[2]$, which we could denote

$$\text{Hom}_{\boldsymbol{\Delta}}([1], [2]) = \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2)\}.$$

It may seem strange, but the category $\boldsymbol{\Delta}^{\text{op}}$ turns out to be quite useful in algebraic topology. It is the indexing category for a combinatorial approach to the homotopy theory of spaces. That is, we can represent something like the category of spaces and continuous maps using the functor category $\textbf{sSet} := \text{Fun}(\boldsymbol{\Delta}^{\text{op}}, \textbf{Set})$, which is called the *category of simplicial sets.*

This may seem very complicated compared to something we did earlier, namely simplicial complexes. But simplicial sets have excellent formal properties that simplicial complexes do not. We will not go further with this here, but through the work of Dan Kan, André Joyal, Jacob Lurie, and many others, simplicial sets have allowed category theory to pierce deeply into the realm of topology and vice versa.

4.6.2 Grothendieck construction

Let $\mathcal{C}$ be a database schema (or category) and let $J \colon \mathcal{C} \to \textbf{Set}$ be an instance. We have been drawing this in table form, but there is another standard way of laying out the data in J, called the *resource descriptive framework* or RDF. Developed for the web, RDF is a useful format when one does not have a schema in hand, e.g. when scraping information off of a website, one does not know what schema will be best. In these cases, information is stored in so-called RDF triples, which are of the form

$$\langle \text{Subject, Predicate, Object} \rangle$$

For example, one might see something like

Subject	**Predicate**	**Object**
A01	occurredOn	D13114
A01	performedBy	P44
A01	actionDescription	Told congress to raise debt ceiling
D13114	hasYear	2013
D13114	hasMonth	January
D13114	hasDay	14
P44	FirstName	Barack
P44	LastName	Obama

(4.24)

Category-theoretically, it is quite simple to convert a database instance $J\colon \mathcal{C} \to \mathbf{Set}$ into an RDF triple store. To do so, we use the *Grothendieck construction*, which is more aptly named the category of elements construction, defined below.[32]

Definition 4.6.2.1. Let $\mathcal{C}$ be a category and let $J\colon \mathcal{C} \to \mathbf{Set}$ be a functor. The *category of elements of* J, denoted $\int_{\mathcal{C}} J$, is defined as follows:

$$\begin{aligned} \mathrm{Ob}(\int_{\mathcal{C}} J) &:= \{(C,x) \mid C \in \mathrm{Ob}(\mathcal{C}), x \in J(C)\}. \\ \mathrm{Hom}_{\int_{\mathcal{C}} J}((C,x),(C',x')) &:= \{f\colon C \to C' \mid J(f)(x) = x'\}. \end{aligned}$$

There is a natural functor $\pi_J\colon \int_{\mathcal{C}} J \longrightarrow \mathcal{C}$. It sends each object $(C,x) \in \mathrm{Ob}(\int_{\mathcal{C}} J)$ to the object $C \in \mathrm{Ob}(\mathcal{C})$. And it sends each morphism $f\colon (C,x) \to (C',x')$ to the morphism $f\colon C \to C'$. We call π_J the *projection functor*.

Example 4.6.2.2. Let A be a set, and consider it as a discrete category. We saw in Exercise 4.3.3.4 that a functor $S\colon A \to \mathbf{Set}$ is the same thing as an A-indexed set, as discussed in Section 2.7.6.10. We will follow Definition 2.7.6.12 and for each $a \in A$ write $S_a := S(a)$.

What is the category of elements of a functor $S\colon A \to \mathbf{Set}$? The objects of $\int_A S$ are pairs (a,s) where $a \in A$ and $s \in S(a)$. Since A has nothing but identity morphisms, $\int_A S$ has nothing but identity morphisms; i.e. it is the discrete category on a set. In fact that set is the disjoint union

$$\int_A S = \bigsqcup_{a \in A} S_a.$$

The functor $\pi_S\colon \int_A S \to A$ sends each element in S_a to the element $a \in A$.

[32] Apparently, Alexander Grothendieck did not invent this construction, it was discussed prior to Grothendieck's use of it, e.g. by Mac Lane. But more to the point, the term Grothendieck construction is not grammatically suited in the sense that both the following are awkward in English: "the Grothendieck construction of J is ..." (awkward because J is not being constructed but used in a construction) and "the Grothendieck construct for J is..." (awkward because it just is). The term *category of elements* is more descriptive and easier to use grammatically.

One can see this as a kind of histogram. For example, let $A = \{\texttt{BOS}, \texttt{NYC}, \texttt{LA}, \texttt{DC}\}$ and let $S \colon A \to \mathbf{Set}$ assign

$$\begin{aligned} S_{\texttt{BOS}} &= \{\texttt{Abby}, \texttt{Bob}, \texttt{Casandra}\}, \\ S_{\texttt{NYC}} &= \varnothing, \\ S_{\texttt{LA}} &= \{\texttt{John}, \texttt{Jim}\}, \text{and} \\ S_{\texttt{DC}} &= \{\texttt{Abby}, \texttt{Carla}\}. \end{aligned}$$

Then the category of elements of S would look like the (discrete) category at the top:

$$\begin{tikzcd}
\int_A S = & \overset{(\texttt{BOS},\texttt{Abby})}{\bullet} & & & \\
& \overset{(\texttt{BOS},\texttt{Bob})}{\bullet} & & \overset{(\texttt{LA},\texttt{John})}{\bullet} & \overset{(\texttt{DC},\texttt{Abby})}{\bullet} \\
& \overset{(\texttt{BOS},\texttt{Casandra})}{\bullet} & & \overset{(\texttt{LA},\texttt{Jim})}{\bullet} & \overset{(\texttt{DC},\texttt{Carla})}{\bullet} \\
& & {} \arrow[d, "\pi_S"'] & & \\
A = & \overset{\texttt{BOS}}{\bullet} & \overset{\texttt{NYC}}{\bullet} & \overset{\texttt{LA}}{\bullet} & \overset{\texttt{DC}}{\bullet}
\end{tikzcd} \tag{4.25}$$

We also see that the category of elements construction has converted an A-indexed set into a relative set over A, as in Definition 2.7.6.7.

The above example does not show at all how the Grothendieck construction transforms a database instance into an RDF triple store. The reason is that our database schema was A, a discrete category that specifies no connections between data (it simply collects the data into bins). So lets examine a more interesting database schema and instance. This is taken from [Sp2].

Application 4.6.2.3. Consider the schema below, which we first encountered in Example 3.5.2.1:

$$\mathcal{C} := \boxed{\begin{array}{c} \underline{\texttt{Employee}\text{ manager worksIn} \simeq \texttt{Employee}\text{ worksIn}} \\ \underline{\texttt{Department}\text{ secretary worksIn} \simeq \texttt{Department}} \\ \begin{tikzcd}
& \overset{\texttt{Employee}}{\bullet} \arrow[loop above, "\text{manager}"] \arrow[dl, "\text{first}"'] \arrow[dr, "\text{last}"] \arrow[rr, shift left, "\text{worksIn}"] & & \overset{\texttt{Department}}{\bullet} \arrow[ll, shift left, "\text{secretary}"] \arrow[d, "\text{name}"] \\
\overset{\texttt{FirstNameString}}{\bullet} & & \overset{\texttt{LastNameString}}{\bullet} & \overset{\texttt{DepartmentNameString}}{\bullet}
\end{tikzcd} \end{array}} \tag{4.26}$$

And consider the instance $J\colon \mathcal{C} \to \mathbf{Set}$, which we first encountered in (3.13) and (3.15)

Employee				
ID	**first**	**last**	**manager**	**worksIn**
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Emmy	Noether	103	q10

Department		
ID	**name**	**secretary**
q10	Sales	101
x02	Production	102

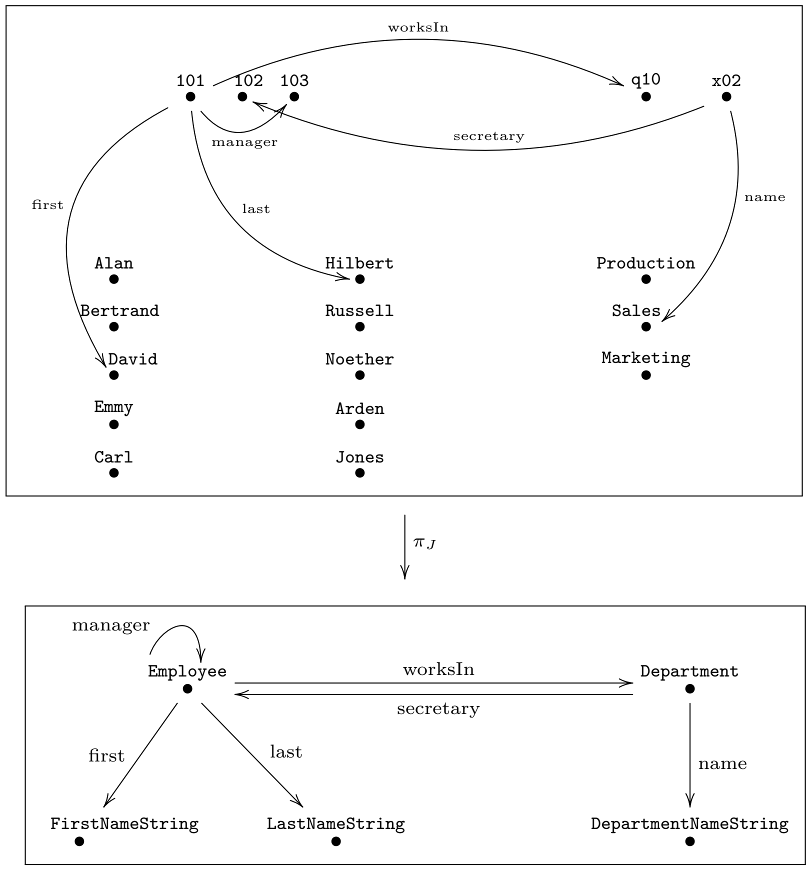
FirstNameString
ID
Alan
Bertrand
Carl
David
Emmy

LastNameString
ID
Arden
Hilbert
Jones
Noether
Russell

DepartmentNameString
ID
Marketing
Production
Sales

The category of elements of $J\colon \mathcal{C} \to \mathbf{Set}$ looks like this:

$$\int_{\mathcal{C}} J = \qquad (4.27)$$



$$\mathcal{C} =$$

In the above drawing (4.27) of $\int_{\mathcal{C}} J$, we left out 10 arrows for ease of readability, for example, we left out an arrow $\overset{\texttt{102}}{\bullet} \xrightarrow{\texttt{first}} \overset{\texttt{Bertrand}}{\bullet}$.

For the punchline, how do we see the category of elements $\int_{\mathcal{C}} J$ as an RDF triple store? For each arrow in $\int_{\mathcal{C}} J$, we take the triple consisting of the source vertex, the arrow name, and the target vertex. So our triple store would include triples such as $\langle$`102 first Bertrand`$\rangle$ and $\langle$`101 manager 103`$\rangle$.

◊◊

Exercise 4.6.2.4. Come up with a schema and instance whose category of elements contains (at least) the data from (4.24). ◊

Slogan 4.6.2.5.

> " *The Grothendieck construction takes structured, boxed-up data and flattens it by throwing it all into one big space. The projection functor is then tasked with remembering which box each datum originally came from.* "

Exercise 4.6.2.6. Recall from Section 3.1.2.10 that a finite state machine is a free monoid $(\mathrm{List}(\Sigma), [\ ], +\!\!+)$ acting on a set X. Recall also that we can consider a monoid as a category $\mathcal{M}$ with one object and a monoid action as a set-valued functor $F\colon \mathcal{M} \to \mathbf{Set}$, (see Section 4.2.1.1). In the case of Figure 3.1 the monoid in question is $\mathrm{List}(a, b)$, which can be drawn as the schema

$$\boxed{\begin{tikzcd} \blacktriangle \arrow[loop left, "a"] \arrow[loop right, "b"] \end{tikzcd}}$$

and the functor $F\colon \mathcal{M} \to \mathbf{Set}$ is recorded in an action table in Example 3.1.3.1. What is $\int_{\mathcal{M}} F$? How does it relate to the picture in Figure 3.1? ◊

4.6.3 Full subcategory

Definition 4.6.3.1. Let $\mathcal{C}$ be a category and let $X \subseteq \mathrm{Ob}(\mathcal{C})$ be a set of objects in $\mathcal{C}$. The *full subcategory of* $\mathcal{C}$ *spanned by* X is the category, which we denote by $\mathcal{C}_{\mathrm{Ob}=X}$, with objects $\mathrm{Ob}(\mathcal{C}_{\mathrm{Ob}=X}) := X$ and with morphisms $\mathrm{Hom}_{\mathcal{C}_{\mathrm{Ob}=X}}(x, x') := \mathrm{Hom}_{\mathcal{C}}(x, x')$.

Example 4.6.3.2. The following are examples of full subcategories. We will name them in the form "X inside of Y", and each time we mean that X and Y are names of categories, the category X can be considered as a subcategory of the category Y in some sense, and it is full. In other words, all morphisms in Y "count" as morphisms in X.

- Finite sets inside of sets, $\mathbf{Fin} \subseteq \mathbf{Set}$;
- Finite sets of the form $\underline{n}$ inside of $\mathbf{Fin}$;
- Linear orders of the form $[n]$ inside of all finite linear orders, $\mathbf{\Delta} \subseteq \mathbf{FLin}$;
- Groups inside of monoids, $\mathbf{Grp} \subseteq \mathbf{Mon}$;
- Monoids inside of categories, $\mathbf{Mon} \subseteq \mathbf{Cat}$;
- Sets inside of graphs, $\mathbf{Set} \subseteq \mathbf{Grph}$;
- Partial orders (resp. linear orders) inside of $\mathbf{PrO}$;

- Discrete categories (resp. indiscrete categories) inside of **Cat**;

Remark 4.6.3.3. A subcategory $\mathcal{C} \subseteq \mathcal{D}$ is (up to isomorphism) just a functor $i\colon \mathcal{C} \to \mathcal{D}$ that happens to be injective on objects and arrows. The subcategory is full if and only if i is a full functor in the sense of Definition 4.3.4.12.

Example 4.6.3.4. Let $\mathcal{C}$ be a category, let $X \subseteq \mathrm{Ob}(\mathcal{C})$ be a set of objects, and let $\mathcal{C}_{\mathrm{Ob}=X}$ denote the full subcategory of $\mathcal{C}$ spanned by X. We can realize this as a fiber product of categories. Indeed, recall that for any set, we can form the indiscrete category on that set; see Example 4.3.4.3. In fact, we have a functor $Ind\colon \mathbf{Set} \to \mathbf{Cat}$. Thus our function $X \to \mathrm{Ob}(\mathcal{C})$ can be converted into a functor between indiscrete categories $Ind(X) \to Ind(\mathrm{Ob}(\mathcal{C}))$. There is also a functor $\mathcal{C} \to Ind(\mathrm{Ob}(\mathcal{C}))$ sending each object to itself. Then the full subcategory of $\mathcal{C}$ spanned by X is the fiber product of categories,

$$\begin{tikzcd}
\mathcal{C}_{\mathrm{Ob}=X} \arrow[r] \arrow[d] & \mathcal{C} \arrow[d] \\
Ind(X) \arrow[r] & Ind(\mathrm{Ob}(\mathcal{C}))
\end{tikzcd}$$

Exercise 4.6.3.5. Including all identities and all compositions, how many morphisms are there in the full subcategory of **Set** spanned by the objects $\{\underline{0}, \underline{1}, \underline{2}\}$? Write them out. ◊

4.6.4 Comma categories

Category theory includes a highly developed and interoperable catalogue of materials and production techniques. One such is the comma category.

Definition 4.6.4.1. Let $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ be categories and let $F\colon \mathcal{A} \to \mathcal{C}$ and $G\colon \mathcal{B} \to \mathcal{C}$ be functors. The *comma category of $\mathcal{C}$ morphisms from F to G*, denoted $(F \downarrow_{\mathcal{C}} G)$ or simply $(F \downarrow G)$, is the category with objects

$$\mathrm{Ob}(F \downarrow G) = \{(a, b, f) \mid a \in \mathrm{Ob}(\mathcal{A}), b \in \mathrm{Ob}(\mathcal{B}), f\colon F(a) \to G(b) \text{ in } \mathcal{C}\}$$

and for any two objects (a, b, f) and (a', b', f') the set $\mathrm{Hom}_{(F\downarrow G)}((a, b, f), (a', b', f'))$ of morphisms $(a, b, f) \longrightarrow (a', b', f')$ is

$$\{(q, r) \mid q\colon a \to a' \text{ in } \mathcal{A},\ \ r\colon b \to b' \text{ in } \mathcal{B},\ \text{ such that } f' \circ F(q) = G(r) \circ f\}.$$

In pictures,

$$\mathrm{Hom}_{(F\downarrow G)}((a,b,f),(a',b',f')) := \left\{ \begin{tikzcd}
a \arrow[d, "q"'] & F(a) \arrow[r, "f"] \arrow[d, "F(q)"'] \arrow[dr, phantom, "\checkmark"] & G(b) \arrow[d, "G(r)"] & b \arrow[d, "r"] \\
a' & F(a') \arrow[r, "f'"'] & G(b') & b'
\end{tikzcd} \right\}$$

We refer to the diagram $\mathcal{A} \xrightarrow{F} \mathcal{C} \xleftarrow{G} \mathcal{B}$ (in **Cat**) as the *setup* for the comma category $(F \downarrow G)$.

There is a canonical functor $(F \downarrow G) \to \mathcal{A}$ called *left projecton*, sending (a, b, f) to a, and a canonical functor $(F \downarrow G) \to \mathcal{B}$ called *right projection*, sending (a, b, f) to b.

A setup $\mathcal{A} \xrightarrow{F} \mathcal{C} \xleftarrow{G} \mathcal{B}$ is reversable; i.e. we can flip it to obtain $\mathcal{B} \xrightarrow{G} \mathcal{C} \xleftarrow{F} \mathcal{A}$. However, note that $(F \downarrow G)$ is different than (i.e. almost never equivalent to) $(G \downarrow F)$, unless every arrow in $\mathcal{C}$ is an isomorphism.

Slogan 4.6.4.2.

> " *When two categories $\mathcal{A}, \mathcal{B}$ can be interpreted in a common setting $\mathcal{C}$, the comma category integrates them by recording how to move from $\mathcal{A}$ to $\mathcal{B}$ inside $\mathcal{C}$.* "

Example 4.6.4.3. Let $\mathcal{C}$ be a category and $I\colon \mathcal{C} \to \mathbf{Set}$ a functor. In this example we show that the comma category construction captures the notion of taking the category of elements $\int_{\mathcal{C}} I$; see Definition 4.6.2.1.

Consider the set $\underline{1}$, the category $Disc(\underline{1})$, and the functor $F\colon Disc(\underline{1}) \to \mathbf{Set}$ sending the unique object to the set $\underline{1}$. We use the comma category setup $\underline{1} \xrightarrow{F} \mathbf{Set} \xleftarrow{I} \mathcal{C}$. There is an isomorphism of categories

$$\int_{\mathcal{C}} I \cong (F \downarrow I).$$

Indeed, an object in $(F \downarrow I)$ is a triple (a, b, f) where $a \in \mathrm{Ob}(\underline{1}), b \in \mathrm{Ob}(\mathcal{C})$, and $f\colon F(a) \to I(b)$ is a morphism in **Set**. There is only one object in $\underline{1}$, so this reduces to a pair (b, f) where $b \in \mathrm{Ob}(\mathcal{C})$ and $f\colon \{☺\} \to I(b)$. The set of functions $\{☺\} \to I(b)$ is isomorphic to $I(b)$, as we saw in Exercise 2.1.2.14. So we have reduced $\mathrm{Ob}(F \downarrow I)$ to the set of pairs (b, x) where $b \in \mathrm{Ob}(\mathcal{C})$ and $x \in I(b)$; this is $\mathrm{Ob}(\int_{\mathcal{C}} I)$. Because there is only one function $\underline{1} \to \underline{1}$, a morphism $(b, x) \to (b', x')$ in $(F \downarrow I)$ boils down to a morphism $r\colon b \to b'$ such that the diagram

$$\begin{tikzcd}
\underline{1} \arrow[r, "x"] \arrow[d, equal] & I(b) \arrow[d, "I(r)"] \\
\underline{1} \arrow[r, "x'"'] & I(b')
\end{tikzcd}$$

commutes. But such diagrams are in one-to-one correspondence with the diagrams needed for morphisms in $\int_{\mathcal{C}} I$.

Exercise 4.6.4.4. Let $\mathcal{C}$ be a category and let $c, c' \in \mathrm{Ob}(\mathcal{C})$ be objects. Consider them as functors $c, c'\colon \underline{1} \to \mathcal{C}$, and consider the setup $\underline{1} \xrightarrow{c} \mathcal{C} \xleftarrow{c'} \underline{1}$. What is the comma category $(c \downarrow c')$? ◊

4.6.5 Arithmetic of categories

In Section 2.7.3, we summarized some of the properties of products, coproducts, and exponentials for sets, attempting to show that they lined up precisely with familiar arithmetic properties of natural numbers. Astoundingly, we can do the same for categories.

In the following proposition, we denote the coproduct of two categories $\mathcal{A}$ and $\mathcal{B}$ by the notation $\mathcal{A} + \mathcal{B}$ rather than $\mathcal{A} \sqcup \mathcal{B}$. We also denote the functor category $\mathrm{Fun}(\mathcal{A}, \mathcal{B})$ by $\mathcal{B}^{\mathcal{A}}$. Finally, we use $\underline{0}$ and $\underline{1}$ to refer to the discrete category on 0 and on 1 object, respectively.

Proposition 4.6.5.1. *The following isomorphisms exist for any small categories $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$.*

- $\mathcal{A} + \underline{0} \cong \mathcal{A}$
- $\mathcal{A} + \mathcal{B} \cong \mathcal{B} + \mathcal{A}$
- $(\mathcal{A} + \mathcal{B}) + \mathcal{C} \cong \mathcal{A} + (\mathcal{B} + \mathcal{C})$
- $\mathcal{A} \times \underline{0} \cong \underline{0}$
- $\mathcal{A} \times \underline{1} \cong \mathcal{A}$
- $\mathcal{A} \times \mathcal{B} \cong \mathcal{B} \times \mathcal{A}$
- $(\mathcal{A} \times \mathcal{B}) \times \mathcal{C} \cong \mathcal{A} \times (\mathcal{B} \times \mathcal{C})$
- $\mathcal{A} \times (\mathcal{B} + \mathcal{C}) \cong (\mathcal{A} \times \mathcal{B}) + (\mathcal{A} \times \mathcal{C})$
- $\mathcal{A}^{\underline{0}} \cong \underline{1}$
- $\mathcal{A}^{\underline{1}} \cong \mathcal{A}$
- $\underline{0}^{\mathcal{A}} \cong \underline{0}, \quad \textit{if } \mathcal{A} \neq \underline{0}$
- $\underline{1}^{\mathcal{A}} \cong \underline{1}$
- $\mathcal{A}^{\mathcal{B}+\mathcal{C}} \cong \mathcal{A}^{\mathcal{B}} \times \mathcal{A}^{\mathcal{C}}$
- $(\mathcal{A}^{\mathcal{B}})^{\mathcal{C}} \cong \mathcal{A}^{\mathcal{B}\times\mathcal{C}}$

Proof. These are standard results; see [Mac].

□