# Applied Compositional Thinking for Engineers

You are reading a work-in-progress book created for the class Applied Compositional Thinking for Engineers (ACT4E):

applied-compositional-thinking.engineering

Please visit the website to get the last updated edition.

# Contents

# Part I.

# Category Theory Basics

# 1. Transmutation

## 1.1. Category theory helps reasoning about interfaces and transformations

In engineering design, one creates *systems* out of *components*. Each component has a reason to be in there. We will show how category theory can help in formalizing the chains of causality that underlie a certain design.

We will need to reason at the level of abstraction where we consider the "function", or "functionality", which each component provides, and the "requirements" that are needed to provide the function.

We will start with a simple example of the functioning principle of an electric car.

In an electric car, there is a battery, a store of the electric energy resource. We can see the production of motion as the chain of two transformations:

- The motor transmutes the electric power into rotational motion.
- The rotational motion is converted into translational motion by the wheels and their friction with the road.

We see that there are two types of things in this example:

1. The "transmuters": the motor and wheels.
2. The "transmuted": the electric power , the rotational motion, the translational motion.

For a first qualitative description of the scenario, we might choose to just keep track of what is transmuted into what. We can draw a diagram in which each resource is a point (Fig. 1.1).

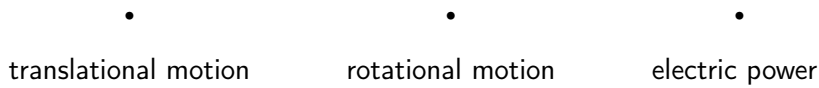



Figure 1.1.: Resources in the electric car example.

Now, we can draw arrows between two points if there is a transmuter between them.

We choose the direction of the arrow such that

$$X \overset{\text{transmuter}}{\longrightarrow} Y \tag{1.1}$$

means that "using transmuter, having $Y$ is sufficient to produce $X$".

> *Remark* 1.1 (Are we going the wrong direction?). The chosen direction for the arrows is completely the opposite of what you would expect if you thought about "input and outputs". There is a good reason to use this convention, though it will be apparent only a few chapters later. In the meantime, it is a good exercise to liberate your mind about the preconception of what an arrow means; in category theory there will be categories where the arrows represent much more abstract concepts than input/output.

Another way to write Eq. (1.1) would be as follows:

$$\mathsf{transmuter} : X \to Y. \tag{1.2}$$

This is now to you something syntactically familiar; when we study the categories of sets and functions between sets we will see that in that context the familiar meaning is also the correct meaning.

With these conventions, we can describe the two transmuters as these arrows:

$$\mathsf{motor} : \mathsf{rotational\ motion} \to \mathsf{electric\ power}, \tag{1.3}$$

$$\mathsf{wheels} : \mathsf{translational\ motion} \to \mathsf{rotational\ motion}. \tag{1.4}$$

We can put these arrows in the diagrams, and obtain the following (Fig. 1.2).

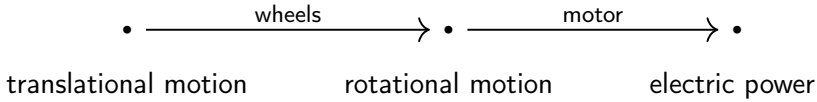



Figure 1.2.: Transmuters are arrows between resources.

In this representation, the arrows are the components of the system. We will learn how to compose these arrows according to the rules of category theory. The basic rule will be *composition*. If we use the semantics that an arrow from resource $X$ to resource $Y$ means "having $Y$ is enough to obtain $X$", then, since $Y$ is enough for $Y$ per definition, we can add a self-loop for each resource. We will call the self-loops *identities* (Fig. 1.3).

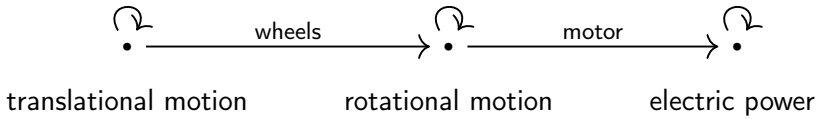



Figure 1.3.: System components and identities.

Furthermore, we might consider the idea of composition of arrows. Suppose that we know that

$$X \xrightarrow{a} Y \qquad \text{and} \qquad Y \xrightarrow{b} Z,$$

that is, using a $b$ we can get a $Y$ from a $Z$, and using an $a$ we can get a $X$ from a $Y$, then we conclude that using and $a$ and a $b$ we can get an $X$ from a $Z$.

In our example, if the arrows wheels and motor exist, then also the arrow "wheels then motor" exists (Fig. 1.4).

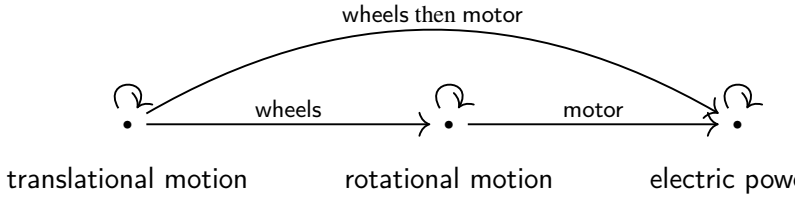



Figure 1.4.: Composition of system components.

So far, we have drawn only one arrow between two points, but we can draw as many as we want. If we want to distinguish between different brands of motors, we would just draw one arrow for each model. For example, Fig. 1.5 shows two models of motors (motor A, and motor B) and two models of wheels (wheels U and wheels V).

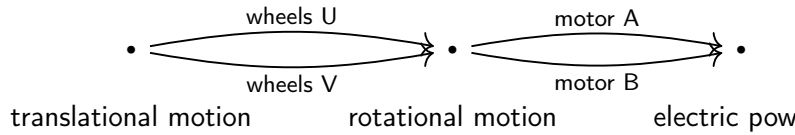



Figure 1.5.: Multiple models for wheels and motors.

The figure implies now the existence of *four* composed arrows: "wheels U then motor A", "wheels U then motor B", "wheels V then motor A", and "wheels V then motor B", all going from translational motion to electric power ;

A "category" is an abstract mathematical structure that captures the properties of these systems of points and arrows and the logic of composition.

The basic terminology is that the points are called **objects**, and the arrows are called **morphisms**.

In our example, the motor and the wheels are the morphisms, and electric power , rotational motion, translational motion are the objects.

Many things can be defined as categories and we will see many examples in this book.

We are now just biding our time before introducing the formal definition of category. At first sight it will be intimidating: there are four parts to the definition, two axioms to

define. Moreover, it is quite a bit technical and it takes half a page to write.

## 1.2. Formal definition of category

The following is the formal definition.

**Definition 1.2** (Category)**.** A *category* $\mathbf{C}$ is specified by four components:

1. **Objects**: a collection[1] $\mathsf{Ob}_{\mathbf{C}}$, whose elements are called *objects*.
2. **Morphisms**: for every pair of objects $X, Y \in \mathsf{Ob}_{\mathbf{C}}$, there is a set $\mathrm{Hom}_{\mathbf{C}}(X, Y)$, elements of which are called *morphisms* from $X$ to $Y$. The set is called the "hom-set from $X$ to $Y$".
3. **Identity morphisms**: for each object $X$, there is an element $\mathrm{id}_X \in \mathrm{Hom}_{\mathbf{C}}(X, X)$ which is called *the identity morphism of* $X$.
4. **Composition rules**: given any morphism $f \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$ and any morphism $g \in \mathrm{Hom}_{\mathbf{C}}(Y, Z)$, there exists a morphism $f \fatsemi g$ in $\mathrm{Hom}_{\mathbf{C}}(X, Z)$ which is the *composition of* $f$ *and* $g$.

Furthermore, the constituents are required to satisfy the following conditions:

1. *Unitality*: for any morphism $f \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$,
$$\mathrm{id}_X \fatsemi f = f = f \fatsemi \mathrm{id}_Y . \tag{1.5}$$
2. *Associativity*: for $f \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$, $g \in \mathrm{Hom}_{\mathbf{C}}(Y, Z)$, and $h \in \mathrm{Hom}_{\mathbf{C}}(Z, W)$,
$$(f \fatsemi g) \fatsemi h = f \fatsemi (g \fatsemi h). \tag{1.6}$$

*Remark* 1.3 (Are we sure we are not going in the wrong direction?). We denote composition of morphisms in a somewhat unusual way–sometimes preferred by category-theorists and computer scientists–namely in *diagrammatic order*.

That is, given $f : X \to Y$ and $g : Y \to Z$, we denote their composite by $(f \fatsemi g) : X \to Z$, pronounced "$f$ then $g$". This is in contrast to the more typical notation for composition, namely $g \circ f$, or simply $gf$, which reads as "$g$ after $f$". The notation $f \fatsemi g$ is sometimes called *infix notation*.

We promise, at some point it will be clear what are the advantages of seemingly doing everything in the wrong direction.

[1] A "collection" is something which may be thought of as a set, but may be "too large" to technically be a set in the formal sense. This distinction is necessary in order to avoid such issues as Russel's paradox.

Note that we may save some ink when drawing diagrams of morphisms:

- We do not need to draw the identity arrows from one object to itself, because, by Definition 1.2, they always exist.
- Given arrows $X \to Y$ and $Y \to Z$, we do not need to draw their composition because, by Definition 1.2, this composition is guaranteed to exist.

With these conventions, we can just draw the arrows motor and wheels in the diagram, and the rest of the diagram is implied (Fig. 1.6).

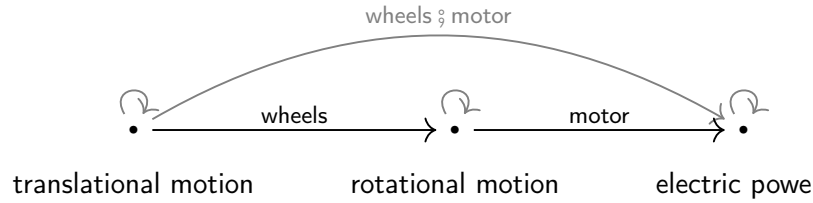



Figure 1.6.: Electric car example. The grey arrows are implied by the properties of a category.

In particular, the electric car example corresponds to the category $\mathbf{C}$ specified by

- *Objects:* $\mathsf{Ob}_{\mathbf{C}} = \{\text{electric power}, \text{rotational motion}, \text{translational motion}\}$.
- *Morphisms*: The system components are the morphisms. For instance, we have motor, wheels, and the morphism wheels ⨟ motor, implied by the properties of the category.

We can slightly expand this example by noting the reverse transformations. In an electric car it is possible to regenerate power; that is, we can obtain rotational motion of the wheels from translational motion (via the morphism move), and then convert the rotational motion into electric power (via the morphism dynamo) (Fig. 1.7, Fig. 1.8).

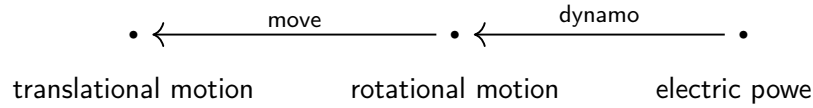



Figure 1.7.: Electric power can be produced from motion.

Given the semantics of the arrows in a category, all compositions of arrows exist, even if they are not drawn explicitly. For example, we can consider the composition wheels ⨟ motor ⨟ dynamo ⨟ move, which converts translational motion into rotational motion, into electric power, then back to rotational motion and translational motion. Note that this is an arrow that has the same head and tail as the identity arrow on translational motion (Fig. 1.9). However, these two arrows are not necessarily the same. In this example we are representing physical systems, so we would in fact not expect them to be the same, since there will be some losses during the many conversions.

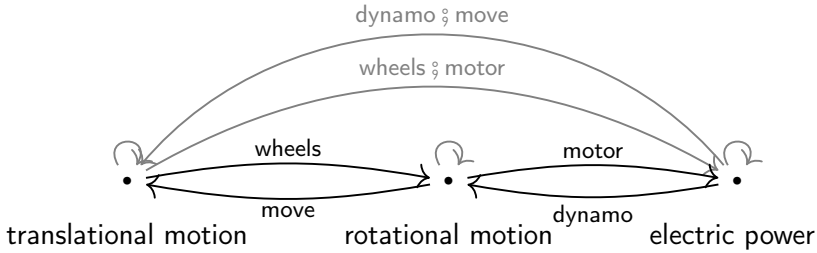



Figure 1.8.: Electric car example: forward and backward transformations.

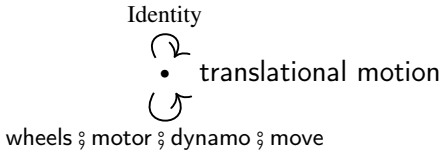



Figure 1.9.: There can be multiple morphisms from an object to itself.

The directionality of the arrows is also important. While the convention of which resource is the tail and which the head is just a typographic convention, it might be the case that we know how to convert one resource into another, but not vice versa. Fig. 1.10 shows an example of a diagram that describes a process which is definitely not invertible.

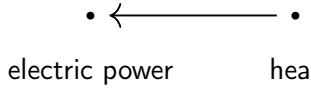



Figure 1.10.: An example of a process which is not invertible.

## 1.3. Currency categories

In this section, we introduce a kind of category for describing currency exchangers. Our idea is to model currencies as objects of a category, and morphisms will describe ways of exchanging between those currencies, e.g., as offered by a currency exchange service.

We start with a set $\mathbf{C}$ of labels for all the currencies we wish to consider, i.e.:

$$\mathbf{C} = \{\mathrm{EUR}, \mathrm{USD}, \mathrm{CHF}, \mathrm{SGD}, ...\}.$$

For each currency $\mathsf{c} \in \mathbf{C}$ we define an object $\mathbb{R} \times \{\mathsf{c}\}$ which represents possible amounts of the given currency $\mathsf{c}$ (we will ignore the issue of rounding currencies to an accuracy

of two decimals places, and we allow negative amounts). The currency label keeps track of which "units" we are using.

Now consider two such objects, say $\mathbb{R} \times \{\mathrm{USD}\}$ and $\mathbb{R} \times \{\mathrm{EUR}\}$. How can we describe the process of changing an amount of USD to an amount of EUR? We model this using two numbers: an exchange rate $a$ and a commission $b$ for the transaction. Given an amount $x \in \mathbb{R}$ of USD, we define a morphism (a currency exchanger) as:

$$E_{a,b} : \mathbb{R} \times \{\mathrm{USD}\} \to \mathbb{R} \times \{\mathrm{EUR}\},$$

by the formula

$$\langle x, \mathrm{USD} \rangle \longmapsto \langle ax - b, \mathrm{EUR} \rangle .$$

Note that that the commission is given in the units of the target currency. Of course, for changing USD to EUR, there may be various different banks or agencies which each offer different exchange rates and/or different commissions. Each of these corresponds to a different morphism from $\mathbb{R} \times \{\mathrm{USD}\}$ to $\mathbb{R} \times \{\mathrm{EUR}\}$.

To build our category, we also need to specify how currency exchangers compose. Given currencies $\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3$, and given currency exchangers

$$E_{a,b} : \mathbb{R} \times \{\mathsf{c}_1\} \to \mathbb{R} \times \{\mathsf{c}_2\} \qquad \text{and} \qquad E_{a',b'} : \mathbb{R} \times \{\mathsf{c}_2\} \to \mathbb{R} \times \{\mathsf{c}_3\}$$

we define the composition $E_{a,b} \fatsemi E_{a',b'}$ to be the currency exchanger

$$E_{aa',a'b+b'} : \mathbb{R} \times \{\mathsf{c}_1\} \to \mathbb{R} \times \{\mathsf{c}_3\}. \tag{1.7}$$

In other words, we compose currency exchangers as one would expect: we multiply the first and the second exchange rates together, and we add the commissions (paying attention to first transform the first commission into the units of the final target currency).

Finally, we also need to specify unit morphisms for our category. These are currency exchangers which "do nothing". For any object $\mathbb{R} \times \{\mathsf{c}\}$, its identity morphism is

$$E_{1,0} : \mathbb{R} \times \{\mathsf{c}\} \to \mathbb{R} \times \{\mathsf{c}\},$$

the currency exchanger with exchange rate "1" and commission "0".

It is a straightforward to check that the composition of currency exchangers as defined above obeys the associative law, and that the identity morphisms act neutrally for composition. Thus we indeed have a category!

*Remark* 1.4. In the above specification of our category of currency exchangers, we can actually just work with the set of currency labels $\mathsf{C}$ as our objects, instead of using "amounts" of the form $\mathbb{R} \times \{\mathsf{c}\}$ as our objects. Indeed, on a mathematical level, the definition of currency exchangers and their composition law (Eq. (1.7)) do not depend on using amounts! Namely, a currency exchanger $E_{a,b}$ is specified by the pair of numbers $\langle a, b \rangle$, and the composition law (1.7) may then, in this

notation, be written as

$$\langle a, b\rangle \fatsemi \langle a', b'\rangle = \langle a'a, a'b + b'\rangle. \tag{1.8}$$

The interpretation is still that currency exchangers change amounts of one currency to amounts in an another currency, but for this we do not need to carry around copies of $\mathbb{R}$ in our notation.

Following the above remark:

**Definition 1.5** (Category **Curr**)**.** A *category of currencies* **Curr** is specified by:

1. *Objects:* a collection of currencies $\mathsf{C}$.
2. *Morphisms:* given two currencies $\mathsf{c}_1, \mathsf{c}_2 \in \mathsf{C}$, morphisms between them are currency exchangers $\langle a, b\rangle$ from $\mathsf{c}_1$ to $\mathsf{c}_2$.
3. *Identity morphism:* given an object $\mathsf{c} \in \mathsf{C}$, its identity morphism is the currency exchanger $\langle 1, 0\rangle$. We also call such morphisms "trivial currency exchangers".
4. *Composition of morphisms:* the composition of morphisms is given by the formula (Eq. (1.8)).

As an illustration, consider three currency exchange companies ExchATM, MoneyLah, and Frankurrencies, which operate on several currencies (Table 1.1).

| Company name | Exchanger label | Direction | $a$ (exchange rate) | $b$ (fixed commission) |
|---|---|---|---|---|
| ExchATM | *A* | USD to CHF | 0.95 CHF/USD | 2.0 CHF |
| ExchATM | *B* | CHF to USD | 1.05 USD/CHF | 1.5 USD |
| ExchATM | *C* | USD to SGD | 1.40 SGD/USD | 1.0 SGD |
| MoneyLah | *D* | USD to CHF | 1.00 CHF/USD | 1.0 CHF |
| MoneyLah | *E* | SGD to USD | 0.72 USD/SGD | 3.0 USD |
| Frankurrencies | *F* | EUR to CHF | 1.20 CHF/EUR | 0.0 CHF |
| Frankurrencies | *G* | CHF to EUR | 1.00 EUR/CHF | 1.0 EUR |

Table 1.1.: Three currency exchange companies operating different currencies.

We can represent this information as a graph, where the nodes are the currencies and the edges are particular exchange operations (Fig. 1.11).

There is a currency category built from the information in Table 1.1 and the graph in Fig. 1.11. Its collection of objects is the set {EUR, USD, CHF, SGD}, and it morphisms are, in total:

- the trivial currency exchanger (identity morphism) $\langle 1, 0\rangle$ for each of the four currencies (which are the objects),
- the currency exchangers corresponding to each item in Table 1.1,
- all possible compositions of the currency exchangers listed in Table 1.1.

The phrase "all possible compositions" is a bit vague. What we mean here can be made more precise. It corresponds to a general recipe for starting with a graph $G$, such as in

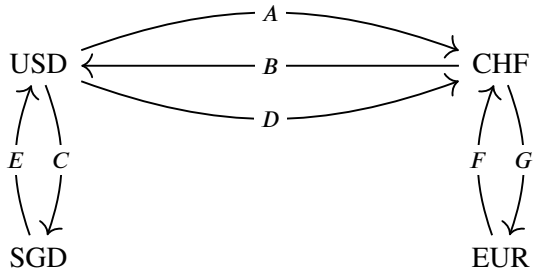



Figure 1.11.: Three currency exchange companies operating different currencies as a graph.

Fig. 1.11, and obtaining from it an associated category, called the *free category on G*. We introduce this concept in the next section.

*Exercise* 1.6 (Temperatures). Define a category of temperature converters, where the objects are Celsius, Kelvin, Fahrenheit, and the morphisms are the rules to transform a measurement from one unit to another.

What relation is there with the currency category?

## 1.4. Definitional impetus vs. computational goals

The category **Curr** represents the set of all possible currency exchangers that could ever exist. However, in this set there would be very irrational agents. For example, there is a currency exchanger that, given 1 USD, will give you back 2 USD; there is one currency exchanger that corresponds to converting USD to CHF back and forth 21 times before getting you the money. There is even one that will not give you back any money.

Moreover, using the composition operations we could produce many more morphisms. In fact, if there are loops, we could traverse the loops multiple times, and, depending on the numbers, finding new morphisms, possibly infinitely many more.

This highlights a recurring topic: often mathematicians will be happy to define a broader category of objects, while, in practice, the engineer will find herself thinking about a more constrained set of objects. In particular, while the mathematician is more concerned with defining categories as hypothetical universes of things, the engineer is typically interested in representing concrete things, and solve some computational problem on the represented structure.

For example, in the case of the currency exchangers, the problem might be that of finding the sequence of the best conversions between a source and a target currency.

First, the engineer would add more constraints to the definition to work with more well-behaved objects. For example, it is reasonable to limit the universe of morphisms in such a way that the action of converting back and forth the same currency to have a cost (through the commission) higher than 0.

In that case, we will find that the optimal paths of currencies never pass through a currency more than once. To see this, consider three currencies $\mathsf{A}, \mathsf{B}, \mathsf{C}$, a currency exchanger $\langle a, b\rangle$ from $\mathsf{A}$ to $\mathsf{B}$ , a currency exchanger $\langle c, d\rangle$ from $\mathsf{B}$ to $\mathsf{C}$ , and a currency exchanger $\langle e, f\rangle$ from $\mathsf{C}$ to $\mathsf{A}$. The composition of the currency exchangers reads:

$$\langle \underbrace{eca}_{g}, \underbrace{ecb + ed + f}_{h} \rangle.$$

Assuming $e = a^{-1}$ (i.e., an exchange rate direction is not more profitable than the other), and $h \neq 0$, because of the commissions one can show that there are multiple morphisms from $\mathsf{A}$ to $\mathsf{A}$, and that the identity morphism is the most "convenient" one. If we only pass through each currency at most once, there are only a finite amount of paths to check, and this might simplify the computational problem.

Second, the engineer might be interested in keeping track only of the "dominant" currency exchangers. For example, if we have two exchangers with the same rate but different commission, we might want to keep track only of the one with the lowest commission.

In the next chapters we will see that there are concepts that will be useful to model these situations:

- There is a concept of *subcategory* that allows to define more specific categories of a parent one, in a way that still satifies the axioms.
- There is a concept called *locally posetal* categories, in which the set of morphisms between two objects is assumed to be a *poset* rather than a *set*, that is, we assume that there is an order, and that this order will be compatible with the operation of composition.

## 1.5. Things that don't matter

In engineering we know that **using the right conventions is essential**.

There are many famous examples of unit mismatches causing disasters or near-disasters:

- The loss of the Mars Climate Orbiter in 1999 was due to the fact that NASA used the metric system, while contractor Lockheed Martin used (by mistake) imperial units.
- In 1983, an Air Canada's Boeing 767 jet ran out of fuel in mid-flight because there was a miscalculation of the fuel needed for the trip. In the end, the pilot managed to succesfully land the "Gimli Glider".

- Going back in history, Columbus wound up in the Bahamas because he miscalculated the Earth's circumference, due to several mistakes, and one of them was assuming that his sources were using the *Roman mile* rather than the *Arabic mile*.[2] Columbus' mathematical mistakes led to a happy incident for him, but not so great outcomes for many others.

However, in category theory, we look at the "essence" of things, and we consider **what is true regardless of conventions**.

Just like this book is written in rather plain English, and could be translated to another language while preserving the meaning, in category theory we look at what is not changed by a 1:1 translation that can be reversed.

This will be covered later in a section on "isomorphisms"; but for now we can look at this in an intuitive way.

### 1.5.1. Typographical conventions don't matter

Some of you might have objected to the conventions that we used in this chapter for the notation for composition of morphisms. We have used the notation $f \fatsemi g$ ("$f$ then $g$") while usually in the rest of mathematics we would have used $g \circ f$ ("$g$ after $f$"). However, any concept we will use is "invariant" to the choice of notation. We can decide to rewrite the book using the other convention and still all the theorems would remain true, and all the falsities will remain false. More technically, we can take any formula written in one convention and rewrite it with the other convention, and viceversa. For example, the formula

$$(f \fatsemi g) \fatsemi h = f \fatsemi (g \fatsemi h)$$

would be transformed in

$$h \circ (g \circ f) = (h \circ g) \circ f.$$

(A bit more advanced category theory can describe this transformation more precisely.)

The same considerations apply for the convention regarding the arrow directions. If we have a category with morphisms such as

$$\mathsf{motor} : \mathsf{rotational\ motion} \to \mathsf{electric\ power}$$

with the semantics of "a motor requires electric power to produce rotational motion", we could define a *different* category, where the conventions are inverted. In this other category, for which we use arrows of different color, we would write

$$\mathsf{motor} : \mathsf{electric\ power} \to \mathsf{rotational\ motion}$$

and the semantics would be "a motor can transmute electric power to rotational motion" (Fig. 1.12).

[2] IEEE Spectrum

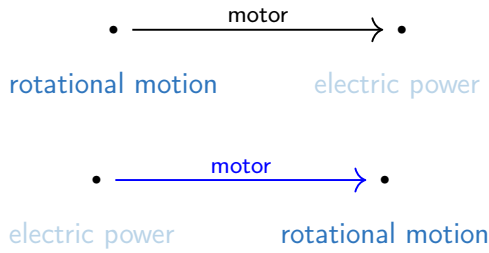



Figure 1.12.: Opposite convention for arrows direction.

These two categories would have the same objects, and the same number of arrows; it's just that the arrows change direction when moving from one category to the other. Intuitively, we would not expect anything substantial to change, because we are just changing a convention. We will see that there is a concept called *opposite category* that formalizes this idea of reversing the direction of the arrows.

### 1.5.2. Diagrams conventions don't matter

Now that we are flexed our isomorphism muscles, we can also talk about the isomorphisms of the visual language.

In engineering, "boxes and wires" diagrams are commonly used to talk about materials transformations and signal flows. In those diagrams one would use boxes to describe the processes and the wires to describe the materials or information that is being transformed. Boxes have "inputs" and "outputs", and arrows have directions representing the causality. From left to right, what is to the left causes what is to the right. The left-to-right directionality seems an utterly obvious choice for most of you who learned languages that are written left-to-right, top-to-bottom as in this book[3].

Fig. 1.13 shows how we would have visually described the first example using the boxes-and-wires conventions. Again, we say that this is just a different convention, because we have a procedure to transform one diagram into the other. This is not as simple as changing the direction of the arrows as in the case of an opposite category. Rather, to go from points-and-arrows to boxes-and-wires:

- Arrows that describe transmuters become boxes that describe processes;
- The points that describe the resources become wires between the boxes.

[3]Note that this paragraph cannot be translated literally to Japanese. It breaks the assumption that we made before, about the fact that we can have a 1:1 literal translation of this book without changing the meaning. You might think that our future hypothetical Japanese translator can make an outstanding job and translate also our figures to go right-to-left, then saying that right-to-left is natural to people that write right-to-left. However, that does not work, because in fact Japanese engineers also use left-to-right diagrams.

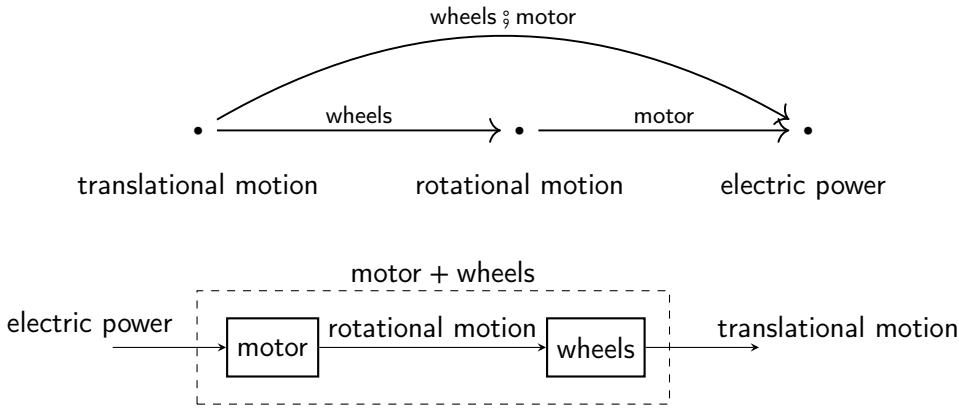



Figure 1.13.: Isomorphisms of resource diagrams.

# 2. Connection

Currency categories illustrated how one can use category theory to think about things transforming into each other. In this section, we want to think about how things connect to each other.

## 2.1. Distribution networks

Consider the type of networks that arise for example in the context of electrical power grids. In a simplified model for a certain region or country, we may have the following kinds of components: power plants (places where electrical power is produced), high voltage transmission lines and nodes, transistor stations, low voltage transmission lines and nodes, and consumers (e.g. homes and businesses). The situation is depicted in Fig. 2.1.

| **Power Plants** | **High Voltage Nodes** | **Low Voltage Nodes** | **Consumers** |
|---|---|---|---|
| Plant 1 | HVN 1 | LVN 1 | C1 |
| | HVN 2 | LVN 2 | C2 |
| Plant 2 | HVN 3 | LVN 3 | C3 |
| | HVN 4 | LVN 4 | C4 |
| Plant 3 | HVN 5 | LVN 5 | C5 |
| | | LVN 6 | C6 |
| | | LVN 7 | |

Figure 2.1.: Components of electrical power grids.

To model the connectivity between the components of the power grid, we now draw arrows between components that are connected. We set the direction of the arrows to flow from energy production, via transmission components, to energy consumption, as depicted in Fig. 2.2.

A possible question one asks about such a power distribution network is: which consumers are serviced by which power sources? For example, some power sources, such as a solar power plant, may fluctuate due to weather conditions, while other power sources, such as a nuclear power plant, may shut down every once in a while due to maintenance work. To see which consumers are connected to which power plants, we

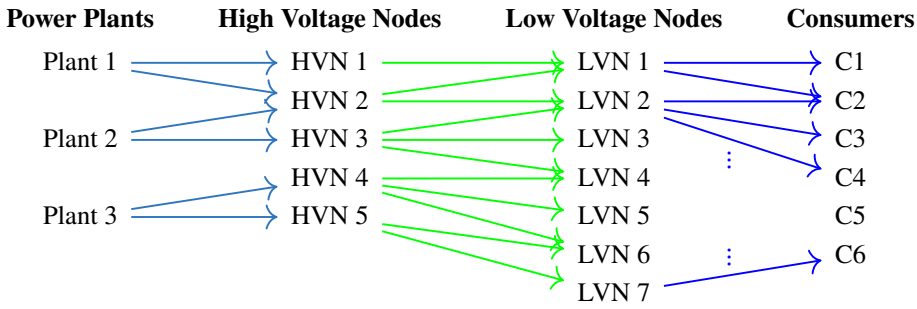



Figure 2.2.: Connectivity between components in electric power grids.

can following paths traced by sequences of arrows, as in Figure Fig. 2.3. There, two possible connectivity paths are depicted (in red and orange, respectively).

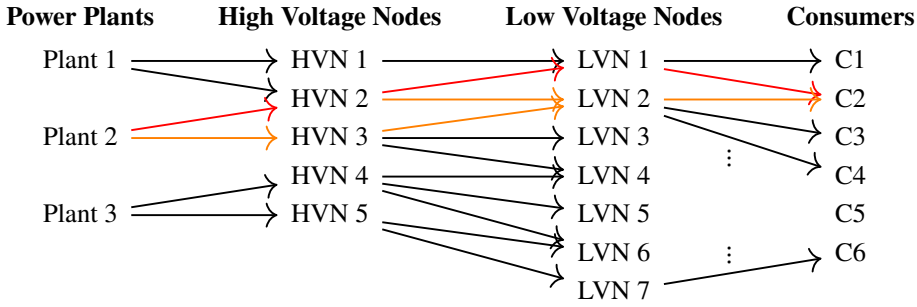



Figure 2.3.: Connection between consumers and power plants.

We also will want to know the overall connectivity structure of transmission lines. For example, some lines may go down during a storm, and we want to ensure enough redundancy in our system. In addition to the connections modeled in Fig. 2.2, we can also include, for example, information about the connectivity of high voltage nodes among themselves, as in Fig. 2.4.

The information encoded in Fig. 2.4 and Fig. 2.2 can also be displayed as a single graph, see Fig. 2.5, Fig. 2.4. If we ignore the directionality of the arrows, this is analogous to a depiction of type shown in Fig. 2.6, which is a schema of a power grid.[1]

[1] See https://en.wikipedia.org/wiki/Electrical_grid and https://doi.org/10.1109/JSYST.2015.2427994

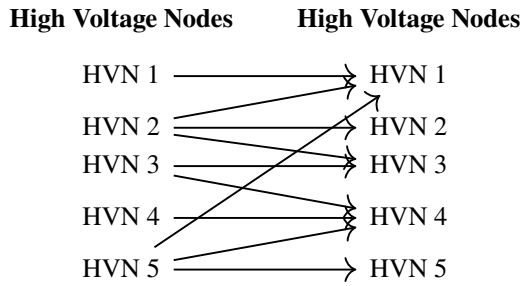



Figure 2.4.: Connectivity between high voltage nodes.

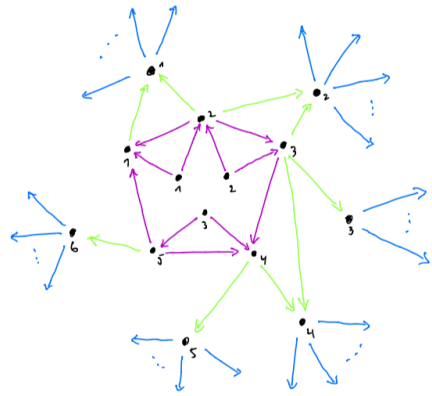


Figure 2.5.: Alternative visualization for connectivity.

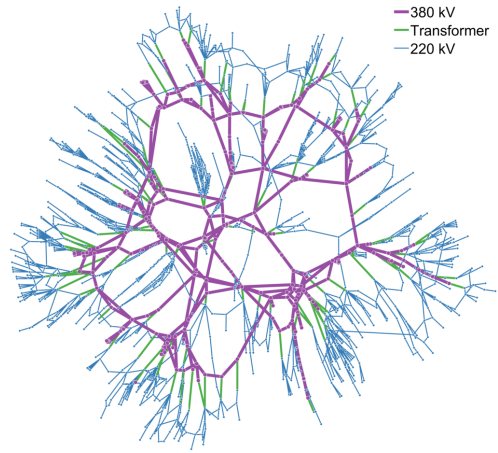



Figure 2.6.: A schematic view of a power grid.

## 2.2. Relations

A basic mathematical notion which underlies the above discussion is that of a **binary relation**.

**Definition 2.1** (Binary relation)**.** A *binary relation* from a set $X$ to a set $Y$ is a subset of the Cartesian product $X \times Y$.

> *Remark* 2.2. We will often drop the word "binary" and simply use the name "relation".

If $X$ and $Y$ are finite sets, we can depict a relation $R \subseteq X \times Y$ graphically as in Fig. 2.7. For each element $\langle x, y\rangle \in X \times Y$, we draw an arrow from $x$ to $y$ if and only if $\langle x, y\rangle \in R \subseteq X \times Y$.

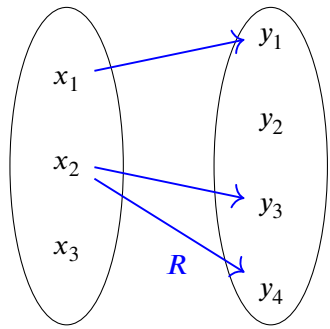



Figure 2.7.

We can also depict this relation graphically as a subset of $X \times Y$ in a "coordinate system way", as in Fig. 2.8. The shaded grey area is the subset $R$ defining the relation.

*Exercise* 2.3. Let $X = Y = \{1, 2, 3, 4\}$ and consider the relation $R \subseteq X \times Y$ defined by

$$R = \{\langle x, y\rangle \in X \times Y \mid x \leq y\}. \tag{2.1}$$

Visualize the relation $R$ via the method in Fig. 2.7 and Fig. 2.8 each.

The visualization in Fig. 2.7 hints at the fact that we can think of a relation $R \subseteq X \times Y$ as a *morphism* from $X$ to $Y$.

**Definition 2.4** (Category **Rel**)**.** The category **Rel** of relations is given by:

1. *Objects*: The objects of this category are all sets.
2. *Morphisms*: Given sets $X, Y$, the homset $\mathbf{Rel}(X, Y)$ consists of all relations $R \subseteq X \times Y$.

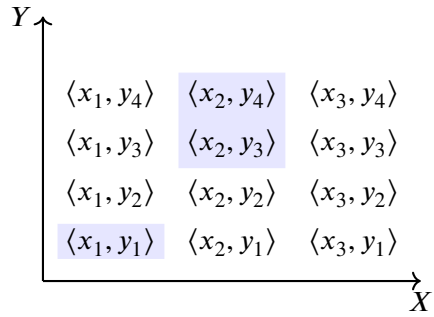



Figure 2.8.: Relations visualized in "coordinate systems".

3. *Identity morphisms*: Given a set $X$, its identity morphism is

$$1_X := \{\langle x, y\rangle \mid x = y\}. \tag{2.2}$$

4. *Composition*: Given relations $R : X \to Y$, $S : Y \to Z$, their composition is given by

$$R \fatsemi S := \{\langle x, z\rangle \mid \exists y \in Y : \ (\langle x, y\rangle \in R) \wedge (\langle y, z\rangle \in S)\}. \tag{2.3}$$

To illustrate the composition rule in Eq. (2.3) for relations, let's consider a simple example, involving sets $X$, $Y$, and $Z$, and relations $R : X \to Y$ and $S : Y \to Z$, as depicted graphically below in Fig. 2.9. Now, according to the rule in Eq. (2.3), the

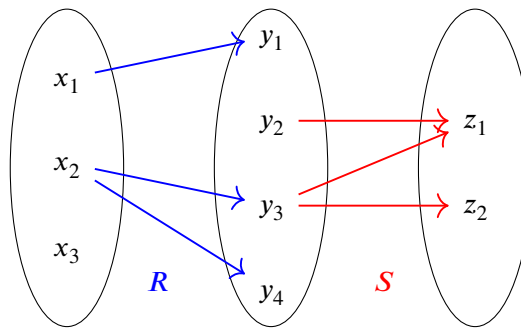



Figure 2.9.: Relations compatible for composition.

composition $R \fatsemi S \subseteq X \times Z$ will be such that $\langle x, z\rangle \in R \fatsemi S$ if and only if there exists some $y \in Y$ such that $\langle x, y\rangle \in R$ and $\langle y, z\rangle \in S$, which, graphically, means that for $\langle x, z\rangle$ to be an element of the relation $R \fatsemi S$, $x$ and $y$ need to be connected by at least one sequence of two arrows such that the target of the first arrow is the source of the second. For example, in Fig. 2.9, there is an arrow from $x_2$ to $y_3$, and from there on

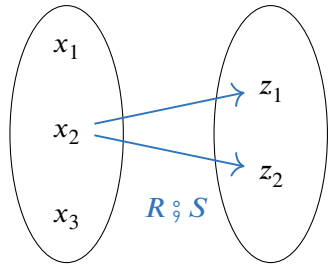



Figure 2.10.: Composition of relations.

to $z_1$, and therefore, in the composition $R \fatsemi S$ depicted in Fig. 2.10, there is an arrow from $x_2$ to $z_1$.

A question on your mind at this point might be: what is the relationship between relations and functions? One point of view is that functions are special kinds of relations.

**Definition 2.5.** Let $X$ and $Y$ be sets. A relation $R \subseteq X \times Y$ is a **function** if it satisfies the following two conditions:

1. $\forall x \in X \quad \exists\, y \in Y : \quad \langle x, y \rangle \in R$
2. $\forall \langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle \in R$ holds : $\quad x_1 = x_2 \Rightarrow y_1 = y_2$.

What does this definition have to do with the "usual" way that we think about functions?

Let start with a relation $R \subseteq X \times Y$ satisfying the conditions of Definition 2.5. We'll build from it a function $f_R : X \to Y$. Choose an arbitrary $x \in X$. According to point 1. in Definition 2.5, there exists a $y \in Y$ such that $\langle x, y \rangle \in R$. So let's choose such a $y$, and call it $f_R(x)$. This gives us recipe to get from any $x$ to a $y$. But maybe you are worried: given a specific $x \in X$, what if we choose $y$ differently each time we apply the recipe? Point 2. guarantees that this can't happen: it says that the element $f_R(x)$ that we associate to a given $x \in X$ is in fact uniquely determined by that $x$. Put another way, the condition 2. says: if $f_R(x_1) \neq f_R(x_2)$, then $x_1 \neq_2$.

Given a function $f : X \to Y$, we can turn it into a relation in a simple way: we consider its graph

$$R_f := \text{graph}(f) = \{\langle x, y \rangle \in X \times Y \mid y = f(x)\}.$$

The relation $R_f$ encodes the same information that $f$ encodes – simply in a different form.

In this text, we take Definition 2.5 as our rigorous definition of a what a function is. Nevertheless, we'll often use functions "in the usual way", e.g. we'll write things like $y = f(x)$.

Another question you may be wondering about is this: if we define functions as special kinds of relations, how then do we define the composition of functions? The answer is that we compose functions simply by the rule for composing relations.

**Lemma 2.6.** Let $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ be relations which are functions. Then their composition $R \fatsemi S \subseteq X \times Z$ is again a function.

*Proof.* We check that $R \fatsemi S$ satisfies the two conditions stated in Definition 2.5.

1. Choose an arbitrary $x \in X$. We need to show that there exists $z \in Z$ such that $\langle x, z\rangle \in R \fatsemi S$. Since $R$ is a function, there exists $y \in Y$ such that $\langle x, y\rangle \in R$. Choose such a $y \in Y$. Then, because $S$ is a function, there exists $z \in Z$ such that $\langle y, z\rangle \in S$. By the definition of composition of relations, we see that $z$ is such that $\langle x, z\rangle \in R \fatsemi S$.
2. Let $\langle x_1, z_1\rangle, \langle x_2, z_2\rangle \in R \fatsemi S$. We need to show that if $x_1 = x_2$, then $z_1 = z_2$. So suppose $x_1 = x_2$. Since $\langle x_1, z_1\rangle, \langle x_2, z_2\rangle \in R \fatsemi S$, there exist $y_1, y_2 \in Y$ such that, respectively,

$$\langle x_1, y_1\rangle \in R \text{ and } \langle y_1, z_1\rangle \in S,$$

$$\langle x_2, y_2\rangle \in R \text{ and } \langle y_2, z_2\rangle \in S.$$

Since $x_1 = x_2$ and $R$ is a function, we conclude that $y_1 = y_2$ must hold. Now, since $S$ is also a function, this implies that $z_1 = z_2$, which is what was to be shown.

□

## 2.3. Path planning

In the section we'll discuss a more "continuum-flavored" (as opposed to "discrete-flavored") example of how one might describe "connectedness" using a category.

Suppose we are planning a hiking tour in the Swiss Alps. In particular, we wish to consider various routes for hikes. We have a map of the relevant region which uses coordinates $\langle x, y, z\rangle$. We assume the $z$-th coordinate is given by an "elevation function", $z = h(x, y)$, and that $h$ is $C^1$, i.e. continuously differentiable. This means that our map of the landscape forms a $C^1$-manifold; let's call it $L$.

We will now define a category where the morphisms are built from $C^1$ paths through the landscape, and such that these paths can be composed, essentially, by concatenation. We take paths which are $C^1$ so that we can speak of the slope (steepness) of a path in any given point, as given by its derivative.

To set things up, we need to have a way to compose $C^1$ paths such that their composition is again $C^1$. For this, the derivative (velocity) at the end of one path must match the starting velocity of the subsequent path.

**Definition 2.7.** Let **Berg** be the category defined as follows:

- Objects are tuples $\langle p, v\rangle$, where
  - $p \in L$,
  - $v \in \mathbb{R}^3$ (we think of this as a tangent vector to $L$ at $p$).
- A morphism $\langle p_1, v_1\rangle \to \langle p_2, v_2\rangle$ is $\langle \gamma, T\rangle$, where
  - $T \in \mathbb{R}_{\geq 0}$,
  - $\gamma : [0, T] \to L$ is a $C^1$ function with $\gamma(0) = p_1$ and $\gamma(p_2)$, as well as $\dot{\gamma}(0) = v_1$ and $\dot{\gamma}(T) = v_2$ (we take one-sided derivatives at the boundaries).
- For any object $\langle p, v\rangle$, we define its identity morphism $1_{\langle p,v\rangle} = \langle \gamma, 0\rangle$ formally: its path $\gamma$ is defined on the closed interval $[0, 0]$, i.e. $T = 0$ and $\gamma(0) = p$. We declare this path to be $C^1$ by convention, and declare its derivative at 0 to be $v$.
- Given morphisms $\langle \gamma_1, T_1\rangle : \langle p_1, v_1\rangle \to \langle p_2, v_2\rangle$ and $\langle \gamma_2, T_2\rangle : \langle p_2, v_2\rangle \to \langle p_3, v_3\rangle$, their composition is $\langle \gamma, T\rangle$ with $T = T_1 + T_2$ and

$$\gamma(t) = \begin{cases} \gamma_1(t) & 0 \leq t \leq T_1 \\ \gamma_2(t - T_1) & T_1 \leq t \leq T_1 + T_2. \end{cases} \tag{2.4}$$

Since we are only amateurs, we don't feel comfortable with hiking on paths that are too steep in some places. We want to only consider paths that have a certain maximum inclination. Mathematically speaking, for any path – as described by a morphism $\langle \gamma, T\rangle$ in the category **Berg**– we can compute its vertical inclination (vertical slope) and renormalize it to give a number in the interval $(-1, 1)$, say. (Here $-1$ represents vertical descent, and 1 represents vertical ascent.) Taking absolute values of inclinations – call the resulting quantity "steepness" – we can compute the maximum steepness that a path $\gamma$ obtains over its domain $[0, T]$. This gives, for every homset $\mathrm{Hom}(\langle p_1, v_1\rangle, \langle p_2, v_2\rangle)$, a function

$$\mathsf{MaxSteepness} : \ \mathrm{Hom}(\langle p_1, v_1\rangle, \langle p_2, v_2\rangle) \longrightarrow [0, 1).$$

Now, suppose we decide that we don't want to traverse paths which have a maximal steepness greater than $1/2$. Paths which satisfy this condition we call *feasible*. Let's consider only the feasible paths in **Berg**. If we keep the same objects as **Berg**, but only consider feasible path, will the resulting structure still form a category? Should we restrict the set of objects for this to be true? We'll let you ponder here; this type of question leads to the notion of a *subcategory*, which we'll introduce soon in a subsequent chapter.

## 2.4. Generating categories from graphs

To begin, we recall some formal definitions related to (directed) graphs.

**Definition 2.8** (Graph)**.** A (directed) *graph* $G = \langle V, A, s, t\rangle$ consists of a set of vertices $V$, a set of arrows $A$, and two functions $s, t : A \to V$, called the *source* and *target*

functions, respectively. Given $a \in A$ with $s(a) = v$ and $t(a) = w$, we say that $a$ is an *arrow* from $v$ to $w$.

> *Remark* 2.9. Both directed graphs and undirected graphs play a prominent role in many kinds of mathematics. In this text, we work primarily with directed graphs and so, from now on, we will drop the "directed": unless indicated otherwise, the word "graph" will mean "directed graph".

**Definition 2.10** (Paths)**.** Let $G$ be a graph. A *path* in $G$ is a sequence of arrows such that the target of one arrow is the source of the next. The *length* of a path is the number of arrows in the sequence. We also formally allow for sequences made up of "zero-many" arrows (such paths therefore have length zero). We call such paths *trivial* or *empty*. If paths describe a journey, then trivial paths correspond to "not going anywhere". The notions of source and target for arrows extend, in an obvious manner, to paths. For trivial paths, the source and target always coincide.

The following definition provides a way of turning any graph into a category.

**Definition 2.11** (Free category on a graph)**.** Let $G = (V, A, s, t)$ be a graph. The *free category on* $G$, denoted $\mathbf{Free}(G)$, has as objects the vertices $V$ of $G$, and given vertices $x \in V$ and $y \in V$, the morphisms $\mathbf{Free}(G)(x, y)$ are the paths from $x$ to $y$. The composition of morphisms is given by concatenation of paths, and for any object $x \in V$, the associated identity morphism $\mathrm{id}_x$ is the trivial path which starts and ends at $x$.

We leave it to the reader to check that the above definition does indeed define a category.

*Exercise* 2.12. Consider the following five graphs. For each graph $G$, how many morphisms in total are there in the associated category $\mathbf{Free}(G)$?

## 2.5. Mobility

For a specific mode of transportation, say a car, we can define a graph

$$G_{\mathrm{c}} = \langle V_{\mathrm{c}}, A_{\mathrm{c}}, s_{\mathrm{c}}, t_{\mathrm{c}} \rangle ,$$

where $V_c$ represents geographical locations which the car can reach and $A_c$ represents the paths it can take (e.g. roads). Similarly, we consider a graph $G_s = \langle V_s, A_s, s_s, t_s \rangle$, representing the subway system of a city, with stations $V_s$ and subway lines going through paths $A_s$, and a graph $G_b = \langle V_b, A_b, s_b, t_b \rangle$, representing onboarding and offboarding at airports. In the following, we want to express intermodality: the phenomenon that someone might travel to a certain intermediate location in a car and then take the subway to reach their final destination.

By considering the graph $G = (V, A, s, t)$ with $V = V_c \cup V_s \cup V_b$ and $A = A_c \cup A_s \cup A_b$, we obtain the desired intermodality graph. Graph $G$ can be seen as a new category, with objects $V$ and morphisms $A$.

**Example 2.13.** Consider the **car** category, describing your road trip in California, with

$$V_c = \{\mathsf{SFO}_c, \mathsf{S.\ Mateo}, \mathsf{Half\ Moon\ Bay}, \mathsf{SBP}_c, \mathsf{Lake\ Balboa}, \mathsf{LAX}_c\},$$

and arrows as in Fig. 2.11. The nodes represent typical touristic road-trip checkpoints in California and the arrows represent famous highways connecting them.

$\mathsf{SFO}_c$ – US101 → S. Mateo – CA92 → H. M. Bay — CA1 → $\mathsf{SBP}_c$ · US101S · Lake Balboa — I405 → $\mathsf{LAX}_c$

Figure 2.11.: The **car** category.

Furthermore, consider the **flight** category with $V_f = \{\mathsf{SFO}_f, \mathsf{SJC}, \mathsf{SBP}_f, \mathsf{LAX}_f\}$ and arrows as in Fig. 2.12. The nodes represent airports in california and the arrows represent connections, offerend by specific flight companies.

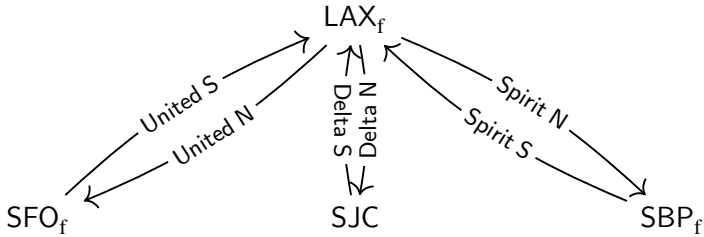



Figure 2.12.: The **flight** category.

We then consider the **board** category, with nodes

$$V_b = \{\mathsf{SFO}_f, \mathsf{SFO}_c, \mathsf{SBP}_f, \mathsf{SBP}_c, \mathsf{LAX}_f, \mathsf{LAX}_c\}$$

and arrows as in Fig. 2.13. Nodes represent airports and airport parkings, and arrows represent the onboarding and offboarding paths one has to walk to get from the parkings to the airport and vice-versa.

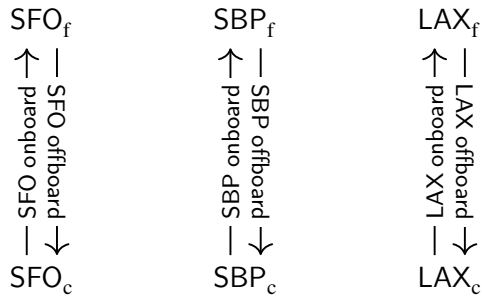



Figure 2.13.: The **board** category.

The combination of the three, which we call the *intermodal graph*, can be represented as a graph, with red arrows for the car network, blue arrows for the flight network, green arrows for the boarding network, and black dashed arrows for intermodal morphisms, arising from composition of morphisms involving multiple modes (Fig. 2.14). Imagine that you are in the parking lot of LAX airport and you want to reach S. Mateo. From there, you will e.g. onboard to a United flight to SFO$_f$, will then offboard reaching the parking lot SFO$_c$, and drive on highway US-101 reaching S. Mateo. This is intermodality.

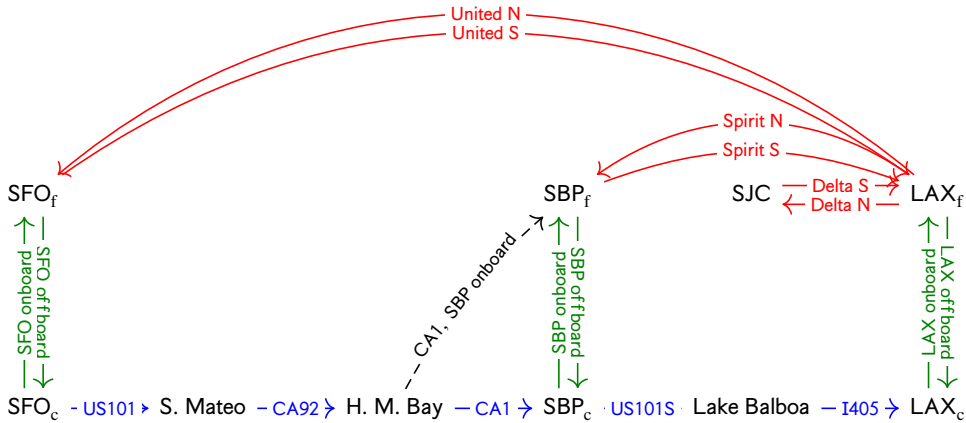



Figure 2.14.: Intermodal graph. The dashed arrows represent intermodal morphisms, and we depict just one of them for simplicity.

The intermodal network category **intermodal** is the free category on the graph illustrated in Fig. 2.14.

## 2.6. Relational Databases

A *relational database* like PostgreSQL, MySQL, etc. presents the data to the user as relations. This does not necessarily mean that the data is stored as tuples, as in the mathematical model, but rather that what the user can do is query and manipulate relations. This conceptual model is now 50 years old.

Can we use the category **Rel** to represent databases [**codd2002relational**]?

Suppose we want to buy an electric stepper motor for a robot that we are building, and for this we consult a catalogue of electric stepper motors[2].

The catalogue might be organized as a large table, where on the left-hand side there is a column listing all available motors (identified with a model ID), and the remaining columns correspond to different attributes that each of the models of motor might have, such as the name of the company that manufactures the motor, the size dimensions, the weight, the maximum power, the price, etc. A simple illustration is provided in Table 2.1.

| Motor ID | Company | Size [mm$^3$] | Weight [g] | Max Power [W] | Cost [USD] |
|---|---|---|---|---|---|
| 1204 | SOYO | 20 x 20 x 30 | 60.0 | 2.34 | 19.95 |
| 1206 | SOYO | 28 x 28 x 45 | 140.0 | 3.00 | 19.95 |
| 1207 | SOYO | 35 x 35 x 26 | 130.0 | 2.07 | 12.95 |
| 2267 | SOYO | 42 x 42 x 38 | 285.0 | 4.76 | 16.95 |
| 2279 | Sanyo Denki | 42 x 42 x 31.5 | 165.0 | 5.40 | 164.95 |
| 1478 | SOYO | 56.4 x 56.4 x 76 | 1,000 | 8.96 | 49.95 |
| 2299 | Sanyo Denki | 50 x 50 x 16 | 150.0 | 5.90 | 59.95 |

Table 2.1.: A simplified catalogue of motors.

### 2.6.1. Can we use Rel for relational databases?

A database table can be seen as representing an $n$-ary relation with $n = 7$, as we are expressing a relation over the sets

$$M \times C \times S \times W \times J \times P,$$

where $M$ represents the set of motor IDs, $C$ the set of companies producing motors, $S$ the set of motor sizes, $W$ the set of motor weights, $J$ the set of possible maximal powers, and $P$ the set of possible prices. An $n$-ary relation is a relation over $n$ sets, just like a binary relation is a relation over 2 sets.

**Definition 2.14** ($n$-ary relation)**.** An $n$-ary relation on $n$ sets $\langle X_1, X_2, \dots, X_n \rangle$ is a subset of the product set

$$X_1 \times X_2 \times \cdots \times X_n. \tag{2.5}$$

[2]See pololu.com for a standard catalogue of electric stepper motors.

**Rel** only allows binary relations. Morphisms in **Rel** have 1 source and 1 target. There is no immediate and natural way to represent $n$-ary relations using **Rel**.

To represent relational databases categorically, there are at least 3 options.

**Option 1: Hack it** We will introduce the notion of *products* and *isomorphisms*. This will allow us to say that because

$$X_1 \times X_2 \times X_3 \times \cdots \times X_n,$$

is isomorphic to

$$X_1 \times (X_2 \times X_3 \cdots \times X_n)$$

we can talk about $n$-ary relations in terms of binary relations. This is not really a natural way to do it.

**Option 2: Mutant Morphisms** What if morphisms could have more than "two legs"? There are indeed theories that work with more complicated arrows. For example: multicategories, polycategories, operads.

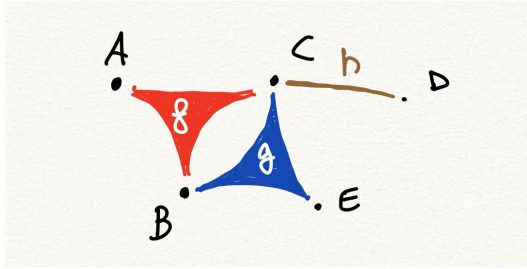



Figure 2.15.: Can you imagine how to define composition with mutant morphisms with more than two legs?

**Option 3: Categorical databases** A different perspective is that of *categorical databases* [**spivak2019categorical**]. In this modeling framework one does not model the data tables as relations directly. Rather the data is described as a functor from a category representing the schema to **Set**.

# 3. Specialization

## 3.1. Databases, sets, functions

We continue the discussion of Section 2.6.

In the particular case of tables with primary keys, things are easier. In relational databases, a table column is a primary key if the values of that column are guaranteed to be unique.

If the values in the column are unique, the column serves as the name of the row. In the table above the motor ID serves as the primary key.

Consider a table with columns $P \times X_1 \times X_2 \times \cdots X_n$, where $P$ is the primary key column. Then, given a key $p \in P$, we can obtain the value in the other columns. We first find the unique row with the key $p$, and then we read out the values.

Therefore, a table with columns $P \times X_1 \times X_2 \times \cdots X_n$ can be seen as a tuple $\langle S, \{f_i\}_i \rangle$:

- A subset $A \subseteq P$ that gives us the available keys.
- $n$ read-out functions $f_i : P \to X_i$, each giving the corresponding value of the $i$-th attribute.

In this example, we can consider the primary key to be the set

$$M := \{1204, 1206, 1207, 2267, 2279, 1478, 2299\},$$

of models of motors. The other colums are given by the set

$$C := \{\mathsf{SOYO}, \mathsf{Sanyo\ Denki}\}$$

of manufacturing companies, the set $S$ of possible motor sizes, the set $W$ of possible weights, the set $J$ of possible maximal powers, and the set $P$ of possible prices. Each attribute of a motor may be thought of as a function from the set $M$ to set of possible values for the given attribute. For example, there is a function $\mathsf{Company} : M \to C$ which maps each model to the corresponding company that manufactures it. So, according to Table 2.1, we have e.g. $\mathsf{Company}(1204) = \mathsf{SOYO}$, and $\mathsf{Company}(2279) = \mathsf{Sanyo\ Denki}$, etc.

Note that in "real life", the catalogue of motors might not have seven entries, as in Table 2.1, but has in fact hundreds of entries, and is implemented digitally as a database, i.e. a collection of interrelated tables. In this case, we will want to be able to search

and filter the data based on various criteria. Many natural operations on tables and databases may be described simply in terms of operations with functions. We will use this setting as a way to introduce compositional aspects of working with sets and functions, and a preview of how this might be useful for thinking, in particular, about databases.

Sticking with Table 2.1, suppose, for instance, that we want to consider only motors from Company Sanyo Denki. In terms of the function

$$\mathsf{Company} : M \to C$$

this corresponds to the preimage $\mathsf{Company}^{-1}(\{\text{Sanyo Denki}\}) = \{2279, 2299\}$, which is a subset of the set $M$. Or, we may want to consider only motors which cost between 40 and 200 USD. In terms of the obvious function

$$\mathsf{Price} : M \to P,$$

this means we wish to restrict ourselves to the preimage

$$\mathsf{Price}^{-1}(\{49.95, 59.95, 164.95\}) = \{1478, 2299, 2279\} \subseteq M.$$

Now suppose we wish to add a column to our table for "volume", because we may want to only consider motors that have, at most, a certain volume. For this we define a set $V$ of possible volumes (let's take $V = \mathbb{R}_{\geq 0}$, the non-negative real numbers), and define a function

$$\begin{aligned} \mathsf{Multiply} : S &\to V \\ \langle l, w, h \rangle &\mapsto l \cdot w \cdot h, \end{aligned}$$

which maps any size of motor to its corresponding volume by multiplying together the given numbers for length, width, and heigth. Now we can compose this function with the function

$$\mathsf{Size} : M \to S$$

to obtain a function

$$\mathsf{Volume} : M \to V,$$

which defines a new column in our table. The composition of functions is usually written as $\mathsf{Volume} = \mathsf{Multiply} \circ \mathsf{Size}$, however we stick to our convention of writing $\mathsf{Volume} = \mathsf{Size} \fatsemi \mathsf{Multiply}$. Schematically, we can represent what we did as a diagram (Fig. 3.1).

We can interpret the arrows in this diagram as being part of a category, one where $M$, $S$, and $V$ are among the objects, and where the functions Size, Multiply and Volume are morphisms. We probably want to consider the other sets associated with our database as also part of this category, and the other functions which we defined so far, too. One idea might be to just include all the sets and functions that we've defined so far, as well as all possible compositions of those functions, and obtain a category, which we

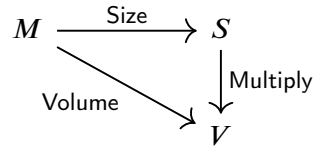



Figure 3.1.: A diagram of functions.

call **Database**, in a way that is similar to how one can build a category from a graph (Section 2.4). This would be an option. However, we may want soon to add new sets and functions to our database framework, or think about new kinds of functions between them that we had not considered before. And we might not want to re-think each time precisely which category we are working with.

## 3.2. The Category Set

A helpful concept here is to think of our specific sets and functions as living in a very (very) large category which contains all possible sets as its objects and all possible functions as its morphisms. This category is known as the category of sets, and it is an important protagonist in category theory. We will denote it by **Set**. It is a short exercise to check that the following does indeed define a category.

**Definition 3.1** (Category of sets)**.** The category of sets **Set** is defined by:

1. *Objects*: all sets.
2. *Morphisms*: given sets $X$ and $Y$, the homset $\mathrm{Hom}_{\mathbf{Set}}(X, Y)$ is the set of all functions from $X$ to $Y$.
3. *Identity morphism*: given a set $X$, its identity morphism $\mathrm{id}_X$ is the identity function $X \to X$, $\mathrm{id}_X(x) = x$.
4. *Composition operation*: the composition operation is the usual composition of functions.

We did say above, however, that we could build a category **Database** which only involves the sets that we are using for our database, and the functions between them that we are working with. What we would need for **Database** to be a category is that if any function is in **Database**, then also its sources and target sets are, and we would need that any composition of functions in **Database** is again in **Database**. (Also, we define the identity morphism for any set in **Database** to be the identity function on that set.) If these conditions are met, **Database** is what is called a *subcategory* of **Set**.

## 3.3. Notion of subcategory

**Definition 3.2** (Subcategory)**.** A subcategory **D** of a category **C** is a category for which:

1. All the objects in $\mathsf{Ob}_{\mathbf{D}}$ are in $\mathsf{Ob}_{\mathbf{C}}$;
2. For any objects $X, Y \in \mathsf{Ob}_{\mathbf{D}}$, $\mathrm{Hom}_{\mathbf{D}}(X, Y) \subseteq \mathrm{Hom}_{\mathbf{C}}(X, Y)$;
3. If $X \in \mathsf{Ob}_{\mathbf{D}}$, then $\mathrm{id}_X \in \mathrm{Hom}_{\mathbf{C}}(X, X)$ is in $\mathrm{Hom}_{\mathbf{D}}(X, X)$ and acts as its identity morphism;
4. If $f : X \to Y$ and $g : Y \to Z$ in **D**, then the composite $f \fatsemi g$ in **C** is in **D** and represents the composite in **D**.

Two important examples of subcategory are the following.

**Example 3.3. FinSet** is the category of finite sets and all functions between them. It is a subcategory of the category **Set** of sets and functions. While an object $X \in \mathsf{Ob}_{\mathbf{Set}}$ is a set with arbitrary cardinality, $\mathsf{Ob}_{\mathbf{FinSet}}$ only includes sets which have finitely many elements. Objects of **FinSet** are in **Set**, but the converse is not true. Furthermore, given $X, Y \in \mathsf{Ob}_{\mathbf{FinSet}}$, we take $\mathrm{Hom}_{\mathbf{FinSet}}(X, Y) = \mathrm{Hom}_{\mathbf{Set}}(X, Y)$.

**Example 3.4.** The category **Set** is a subcategory of **Rel**. To show this, we need to prove the conditions presented in Definition 3.2.

1. In both **Rel** and **Set**, the collection of objects is all sets.
2. Given $X, Y \in \mathsf{Ob}_{\mathbf{Set}}$, we know that $\mathrm{Hom}_{\mathbf{Set}}(X, Y) \subseteq \mathrm{Hom}_{\mathbf{Rel}}(X, Y)$, i.e., that all functions between sets $X, Y$ are a particular subset of all relations between $X, Y$.
3. For each $X \in \mathsf{Ob}_{\mathbf{Set}}$, the identity relation $\mathrm{id}_X = \{\langle x, x' \rangle \in X \times X \mid x = x'\}$ corresponds to the identity function $\mathrm{id}_X : X \to X$ in **Set**.
4. Let $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ be relations which are functions. We need to show that their composition in **Rel**, expressed as $R \fatsemi S \subseteq X \times Z$, is again a function. This was proven in Lemma 2.6.

## 3.4. Drawings

**Definition 3.5.** There exists a category **Draw** in which:

1. An object in $\alpha \in \mathsf{Ob}_{\mathbf{Draw}}$ is a black-and-white drawing, that is a function $\alpha : \mathbb{R}^2 \to \texttt{Bool}$.
2. A morphism in $\mathrm{Hom}_{\mathbf{Draw}}(\alpha, \beta)$ between two drawings $\alpha$ and $\beta$ is an invertible map $f/colon\mathbb{R}^2 \to \mathbb{R}^2$ such that $\alpha(x) = \beta(f(x))$.
3. The identity function at any object $\alpha$ is the identity map on $\mathbb{R}^2$.

4. Composition is given by function composition.

*Exercise* 3.6. Check whether just considering

- affine invertible transformations, or
- rototranslations, or
- scalings, or
- translations, or
- rotations,

as morphisms forms a subcategory of **Draw**.

## 3.5. Other examples of subcategories in engineering

In engineering it is very common to look at specific types of functions; in many cases, the properties of a certain type of function are preserved by function composition, and so they form a category.

### 3.5.1. InjSet forms a subcategory of Set

**Definition 3.7** (Injective function)**.** Let $f : X \to Y$ be a function. The function $f$ is *injective* if, for all $x, x' \in X$ holds: $f(x) = f(x') \implies x = x'$.

**Example 3.8.** We can define a category **InjSet** which has the same objects as **Set** but restricts the morphisms to be *injective functions*. We want to show that **InjSet** is a subcategory of **Set**. Composition and identity morphisms are defined as in **Set**.

Since $\mathsf{Ob}_{\mathbf{InjSet}} = \mathsf{Ob}_{\mathbf{Set}}$, the first condition of Definition 3.2 is satisfied. Injective functions are a particular type of functions: this satisfies the second condition. Given $X \in \mathsf{Ob}_{\mathbf{InjSet}}$, the identity morphism $\mathrm{id}_X \in \mathrm{Hom}_{\mathbf{Set}}(X, X)$ corresponds to the identity morphism in $\mathrm{Hom}_{\mathbf{InjSet}}(X, X)$, i.e., the identity function is injective. This proves the third condition. To check the fourth condition, consider two morphisms $f \in \mathrm{Hom}_{\mathbf{Set}}(X, Y), g \in \mathrm{Hom}_{\mathbf{Set}}(Y, Z)$ such that $f \in \mathrm{Hom}_{\mathbf{InjSet}}(X, Y)$ and $g \in \mathrm{Hom}_{\mathbf{InjSet}}(Y, Z)$. From the injectivity of $f, g$, we know that given $x, x' \in X$, $f(x) = f(x') \Leftrightarrow x = x'$ and $y, y' \in X$, $g(y) = g(y') \Leftrightarrow y = y'$. Furthermore, we have:

$$\begin{aligned}(f \fatsemi g)(x) = (f \fatsemi g)(x') &\implies f(x) = f(x') \\ &\implies x = x',\end{aligned}$$

which proves the fourth condition of Definition 3.2, i.e. that the composition of injective functions is injective.

## 3.6. Subcategories of Berg

Recall the category **Berg** presented in Section 2.3. In the following, we want to give both a positive and a negative example of subcategories related to **Berg**.

We first start our discussion by introducing an *amateur* version of **Berg**, called **BergAma**, which only considers paths (morphisms) in **Berg**, whose steepness does not exceed a critical value, say 1/2. Is **BergAma** a subcategory of **Berg**? Let's check the different conditions:

1. The constraint on the maximum steepness restricts the objects which are acceptable in **BergAma** via the identity morphisms of **Berg**. Indeed, recall that given an object $\langle p, v\rangle \in \mathsf{Ob}_{\mathbf{Berg}}$, the identity morphism is defined as $1_{\langle p,v\rangle} = \langle \gamma, 0\rangle$, with $\gamma(0) = p$ and $\dot{\gamma}(0) = v$. The steepness is computed via $v$. In particular, **BergAma** will only contain objects whose identity morphisms do not exceed the steepness constraint, i.e. $\mathsf{Ob}_{\mathbf{BergAma}} \subseteq \mathsf{Ob}_{\mathbf{Berg}}$.
2. For $A, B \in \mathsf{Ob}_{\mathbf{BergAma}}$, we know that paths satisfying the steepness constraint are specific paths in **Berg**, i.e. $\mathrm{Hom}_{\mathbf{BergAma}} \subseteq \mathrm{Hom}_{\mathbf{Berg}}$.
3. The identity morphisms in **Berg** which satisfy the steepness constraint are, by definition, in **BergAma** and they act as identities there.
4. Given two morphisms $f, g$ which can be composed in **BergAma**, the maximum steepness of their composition $f \fatsemi g$ is given by:

$$\mathrm{MaxSteepness}(f \fatsemi g) = \max\{\mathrm{MaxSteepness}(f), \mathrm{MaxSteepness}(g)\} < 1/2.$$

This shows that **BergAma** is a subcategory of **Berg**. What would an example of non-subcategory of **Berg** be? Let's define a new category **BergLazy**, which now discriminates morphisms based on the lengths of the paths they represent. For instance, assume that as amateur hikers, we don't want to consider morphisms which are more than 1 km long. By concatenating two paths (morphisms) of length 0.6 km in **BergLazy**, the resulting composition will be 1.2 km, violating the posed constraint and hence not being in **BergLazy**. This violates the fourth property of Definition 3.2.

# 4. Sameness

## 4.1. Sameness in category theory

One nice thing about the category of sets is that we are all used to working with sets and functions. And many concepts that are familiar in the setting of sets and functions can actually be reformulated in a way which makes sense for lots of other categories, if not for all categories. It can be fun, and insightful, to see known definitions transformed into "category theory language". For example: the notion of a bijective function is a familiar concept. There are least two ways of saying what it means for a function $f : X \to Y$ of sets to be bijective:

**Definition 1:** "$f : X \to Y$ is bijective if, for every $y \in Y$ there exists precisely one $x \in X$ such that $f(x) = y$;

**Definition 2:** "$f : X \to Y$ is bijective if there exists a function $g : Y \to X$ such that $f \fatsemi g = \mathrm{id}_X$ and $g \fatsemi f = \mathrm{id}_Y$".

It is a short proof to show that the above two definitions are equivalent. The first definition, however, does not lend itself well to generalization in category theory, because it is formulated using something that is very specific to sets: namely, it refers to *elements* of the sets $X$ and $Y$. And we have seen that the objects of a category need not be sets, and so in general we cannot speak of "elements" in the usual sense. Definition 2, on the other hand, can easily be generalized to work in any category. To formulate this version, all we need are morphisms, their composition, the notion of identity morphisms, and the notion of equality of morphisms (for equations such as "$f \fatsemi g = \mathrm{id}_x$"). The generalization we obtain is the fundamental notion of an "isomorphism".

**Definition 4.1** (Isomorphism)**.** Let $\mathbf{C}$ be a category, let $X, Y \in \mathbf{C}$ be objects, and let $f : X \to Y$ be a morphism. We say that $f$ is an **isomorphism** if there exists a morphism $g : Y \to X$ such that $f \fatsemi g = \mathrm{id}_X$ and $g \fatsemi f = \mathrm{id}_Y$.

*Remark* 4.2. The morphism $g$ in the above definition is called the **inverse** of $f$.

> Because of the symmetry in how the definition is formulated, it is easy to see that $g$ is necessarily also an isomorphism, and its inverse is $f$.

*Exercise* 4.3. In Remark Remark 4.2 we wrote *the* inverse. We do this because inverses are in fact unique. Can you prove this? That is, show that if $f : X \to Y$ is an isomorphism, and if $g_1 : Y \to X$ and $g_2 : Y \to X$ are morphisms such that $f \fatsemi g_1 = \mathrm{id}_X$ and $g_1 \fatsemi f = \mathrm{id}_Y$, and $f \fatsemi g_2 = \mathrm{id}_X$ and $g_2 \fatsemi f = \mathrm{id}_Y$, then necessarily $g_1 = g_2$.

**Definition 4.4** (Isomorphic)**.** Let $\mathbf{C}$ be a category, and let $X, Y \in \mathbf{C}$ be objects. We say that $X$ and $Y$ are **isomorphic** if there exists an isomorphism $X \to Y$ or $Y \to X$.

For the formulation of the definition of "isomorphic", mathematicians might often only require the existence of an isomorphism $X \to Y$, say, since by **??** we know there is then necessarily also an isomorphism in the opposing direction, namely the inverse. We choose here the longer, perhaps more cumbersome formulation just to emphasis the symmetry of the term "isomorphic". Also note that the definition leaves unspecified whether there might be just one or perhaps many isomorphisms $X \to Y$.

When two objects are isomorphic, in some contexts we will want to think of them as "the same", and in some contexts we will want to keep track of more information. In fact, in category theory, it is typical to think in terms of different kinds of "sameness". To give a sense of this, let's look at some examples using sets.

**Example 4.5** (Semantic coherence)**.** Suppose Francesca and Gabriel want to share a dish at a restaurant. Francesca only speaks Italian, and Gabriel only speaks German. Let $M$ denote the set of dishes on the menu. For each dish, Francesca can say if she is willing to eat it, or not. This can be modeled by a function $f : M \to \{\mathrm{Si}, \mathrm{No}\}$ which maps a given dish $m \in M$ to the statement "Si" (yes, I'd eat it) or "No" (no, I wouldn't eat it). Gabriel can do similarly, and this can be modeled as a function $g : M \to \{\mathrm{Ja}, \mathrm{Nein}\}$. Then, the subset of dishes of $M$ that both Francesca and Gabriel are willing to eat (and thus able to share) is

$$\{m \in M \mid f(m) = \mathrm{Si} \quad \text{and} \quad g(m) = \mathrm{Ja}\}.$$

Suppose the server at the restaurant knows no Italian and no German. To help with the situation, he introduces a new two-element set: $\{\heartsuit, ☠\}$. Then Francesca and Gabriel can each map their respective positive answers ("Si" and "Ja") to "$\heartsuit$", and their respective negative answers to "☠". This defines isomorphisms

$$\{\mathrm{Si}, \mathrm{No}\} \longleftrightarrow \{\heartsuit, ☠\} \longleftrightarrow \{\mathrm{Ja}, \mathrm{Nein}\}$$

whose compositions provide a translation between the Italian and German two-element sets. Using these isomorphisms, we obtain, by composition, new functions

$$\tilde{f} : M \longrightarrow \{♥, ☠\}, \qquad \tilde{g} : M \longrightarrow \{♥, ☠\},$$

and the set of dishes that Francesca and Gabriel would be willing to share can be written as

$$\{m \in M \mid \tilde{f}(m) = ♥ \quad \text{and} \quad \tilde{g}(m) = ♥\}.$$

This may all seem unnecessarily complicated. The main point of this example is the following. There are infinitely many two-element sets; commonly used ones might be, for example

$$\{0, 1\},\ \{\mathsf{true},\ \mathsf{false}\},\ \{\bot, \top\},\ \{\ \mathsf{left},\ \mathsf{right}\},\ \{-, +\},\ \text{etc.}$$

They are all isomorphic (for any two such sets, there are precisely two possible isomorphisms between them) and we can in principle use any one in place of another. However, in most cases, we should keep precise track of the semantics of what each of the two elements mean in a given context, i.e. how they are being used in interaction with other mathematical constructs.

**Example 4.6** (Relabelling)**.** Consider the little catalogue in Table 2.1. Suppose that your old way of listing models of motors has become outdated and you need to change to a new system, where each model is identified, say, by a unique numerical 10-digit code. Relabelling each of the models with its numerical code corresponds to an isomorphism, say $\mathsf{relabel}$, from the new set $N$ of numerical codes to the old set $M$ of model names. In contrast to the previous example, however, it is of course absolutely necessary to keep track of the isomorphism $\mathsf{relabel}$ that defines the relabelling. This is what holds the information of which code denotes which model.

Note also that all the other labelling functionalities in our example database may be updated by precomposing with $\mathsf{relabel}$. For example, the old "Company" label was described by a function

$$\mathsf{Company} : M \to C.$$

The updated version of the "Company" label, using the new set $N$ of model IDs, is obtained by the composition

$$N \overset{\mathsf{relabel}}{\longrightarrow} M \overset{\mathsf{Company}}{\longrightarrow} C.$$

**Example 4.7.** Going back to currency exchangers, recall that any currency exchanger $E_{a,b}$, given by

$$\begin{aligned} E_{a,b} : \mathbb{R} \times \{\mathrm{USD}\} &\to \mathbb{R} \times \{\mathrm{EUR}\} \\ \langle x, \mathrm{USD}\rangle &\mapsto \langle ax - b, \mathrm{EUR}\rangle \end{aligned}$$

is an isomorphism, since one can define a currency exchanger $E_{a',b'}$ such that

$$E_{a,b} \fatsemi E_{a',b'} = E_{a',b'} \fatsemi E_{a,b} = E_{1,0}.$$

**Example 4.8.** In **FinSet**, isomorphisms from a set to itself are automorphisms, and correspond to *permutations* of the set. Assuming a cardinality of $n$ for the set (i.e., the set has $n$ elements), the number of isomorphisms is given by the number of ways in which one can "rearrange" $n$ elements of the set, which is $n!$.

**Example 4.9.** In **Set**, isomorphisms between $\mathbb{R} \to \mathbb{R}$ correspond to invertible functions.

## 4.2. Isomorphism is not identity

**Example 4.10.** Let's consider currencies, and in particular the sets $\mathbb{R} \times \{\text{USD}\}$ and $\mathbb{R} \times \{\text{USD cents}\}$. These are both objects of the category **Curr** and are isomorphic. Being isomorphic does not mean to be strictly "the same". Indeed, even if the amounts correspond, 10 USD and 1,000 USD cents are different elements of different sets, but there exists an isomorphism between the two. For one direction, the isomorphism transforms USD into USD cents (multiplying the real number by 100); the other direction transforms USD cents into USD (dividing the real number by 100).

# 5. Thinking about trade-offs

So far, the discussion has been purely qualitative. While we discussed how categories can describe the way in which one resource can be turned into another, this kind of modelling did not allow for quantitative statements. For example, it is good to know that we can obtain motion from electric power, but, how fast can we go with a certain amount of power?

To achieve a quantitative theory, we need to specify various degrees of resources and functionality. One way of doing this, is through the idea of partial orders.

## 5.1. Partially ordered sets

Such orderings arise naturally in engineering as criteria for judging whether one design is better or worse than another. As an example, suppose you need to prepare some pizza, i.e., you have to buy specific ingredients and cook them, using a recipe you decide to follow. In this simple example, you can think of having two resources: time and money. A quicker recipe might include more expensive ingredients, and a slower recipe could feature more affordable ones. How to choose among recipes, if you do not prefer one resource over the other? How to model this? In this section, we will assume that functionality and resources are *partially-ordered sets* (*posets*).

**Definition 5.1** (Partially ordered set)**.** A *partially-ordered set* (poset) is a tuple $\langle P, \preceq \rangle$, where $P$ is a set (also called the *carrier set*), together with a relation $\preceq$ on $P$ that is

1. *Reflexive*: For all $x \in P$, $x \preceq x$.
2. *Antisymmetric*: For all $x, y \in P$, if $x \preceq y$ and $y \preceq x$, then $x = y$.
3. *Transitive*: For all $x, y, z \in P$, if $x \preceq y$ and $y \preceq z$, then $x \preceq z$.

A *Hasse diagram* is an economical (in terms of arrows) way to visualize a poset. In a Hasse diagram elements are points, and if $x \preceq y$ then $x$ is drawn lower than $y$ and with an ege connected to it, if no other point is in between. Hasse diagrams are directed graphs.

In the example of the pizza recipes, both time and money can be thought of as partially ordered sets $\left\langle \mathbb{R}_{\geq 0}, \leq \right\rangle$. Imagine that you have recipes costing 1 USD, 2 USD, and 3 USD. This can be represented as in Fig. 5.1.


Figure 5.1.: The cost of pizza ingredients can be represented as a poset.

**Example 5.2.** Consider a poset $P = \{a, b, c, d, e\}$ with $a \preceq b$, $a \preceq c$, $d \preceq c$, and $d \preceq e$. This can be represented with a Hasse diagram as in Fig. 5.2.

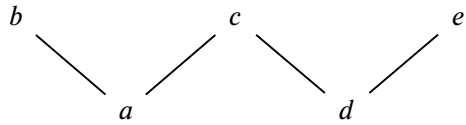



Figure 5.2.: Example of Hasse diagram of $P$.

**Example 5.3** (Singleton poset)**.** If a set has only one element, say $\{1\}$, then there is a unique order relation on it (Fig. 5.3). We denote the resulting poset again by $\{1\}$.

•

Figure 5.3.: The singleton poset.

**Example 5.4.** In this example, we represent all posets up to isomorphisms on up to 4 elements. For one element, one has only the singleton poset (Fig. 5.3). On 2-elements sets, one has the posets reported in Fig. 5.4. On 3-elements sets, one has the posets reported in Fig. 5.5. On 4-elements sets, one has the posets reported in Fig. 5.6.

**Example 5.5** (Booleans)**.** The booleans `Bool` is a poset with carrier set $\{\mathtt{T}, \mathtt{F}\}$ and the order relation given by $b_1 \preceq_{\mathtt{Bool}} b_2$ iff $b_1 \Rightarrow b_2$, that is, $\mathtt{F} \preceq_{\mathtt{Bool}} \mathtt{T}$ (Fig. 5.7).

This relation should be familiar from Table 5.1.

In addition to the operation

$$\Rightarrow : \mathtt{Bool} \times \mathtt{Bool} \to \mathtt{Bool},$$

called *implies*, there are also the familiar *and* ($\wedge$) and *or* ($\vee$) operations. Note that $\wedge$ and $\vee$ are commutative ($b \wedge c = c \wedge b$, $b \vee c = c \vee b$ ), whereas $\Rightarrow$ is not.

Figure 5.4.: All posets on 2-elements sets, up to isomorphisms.

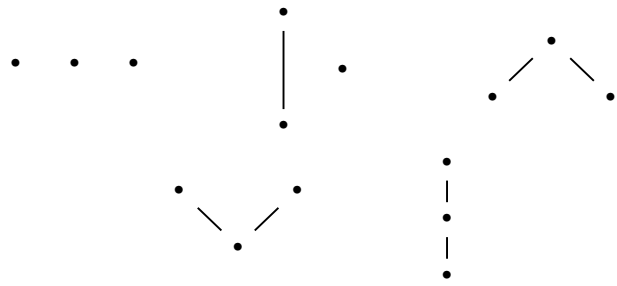


Figure 5.5.: All posets on 3-elements sets, up to isomorphisms.

| $a$ | $b$ | $a \leq_{\texttt{Bool}} b$ | $a \wedge b$ | $a \vee b$ |
|---|---|---|---|---|
| T | T | T | T | T |
| T | F | F | F | T |
| F | T | T | F | T |
| F | F | T | F | F |

Table 5.1.: Properties of the `Bool` poset.

**Example 5.6** (Reals)**.** The real numbers $\mathbb{R}$ form a poset with carrier $\mathbb{R}$ and order relation given by the usual ordering $r_1 \leq r_2$.

**Example 5.7** (Discrete partially ordered sets)**.** Every set $X$ can be considered as a *discrete poset* $\langle X, = \rangle$. Discrete posets are represented as collection of points (Fig. 5.8).

**Example 5.8.** Given a set $X = \{a, b, c\}$, consider its power set $\mathcal{P}(X)$. Define sets as the objects of this new category and define the morphisms to be inclusions (Fig. 5.9). The identity morphism of each set is the inclusion with itself (every set is a subset of itself). Composition is given by composition of inclusions, i.e., if $X \subseteq Y \subseteq Z$, then $X \subseteq Z$.

### A note on preorders

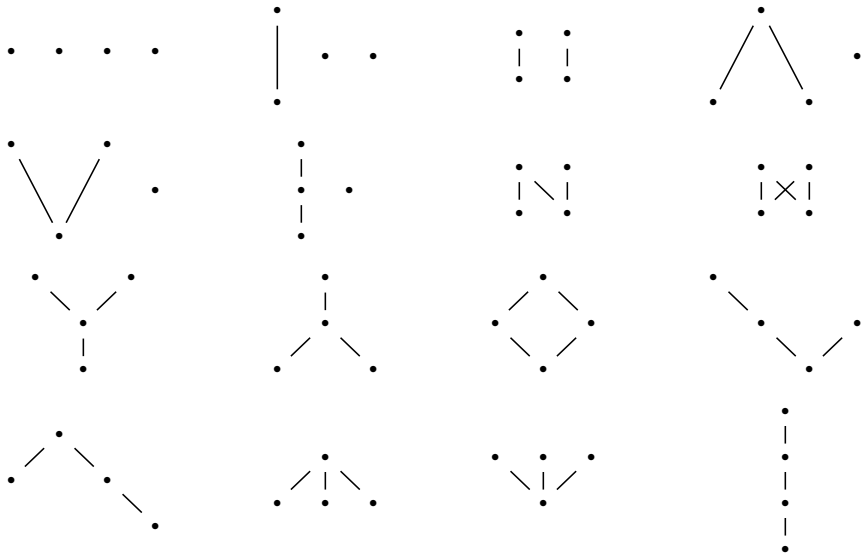


Figure 5.6.: All posets on 4-elements sets, up to isomorphisms.

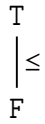



Figure 5.7.

**Definition 5.9** (Preorder)**.** A *preorder* is a tuple $\langle P, \leq \rangle$, where $P$ is a set (also called the *carrier set*), together with a relation $\leq$ on $P$ that is

1. *Reflexive*: For all $x \in P$, $x \preceq x$.
2. *Transitive*: For all $x, y, z \in P$, if $x \preceq y$ and $y \preceq z$, then $x \preceq z$.

The theory of design problems can be easily generalized to preorders. This means that there could be two elements $x$ and $y$ such that $x \preceq y$ and $x \succeq y$ but $x \neq y$ (Fig. 5.10). This is actually common in practice. For example, if the order relation comes from human judgement, such as customer preference, all bets are off regarding the consistency

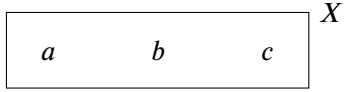



Figure 5.8.: Example of a discrete poset.

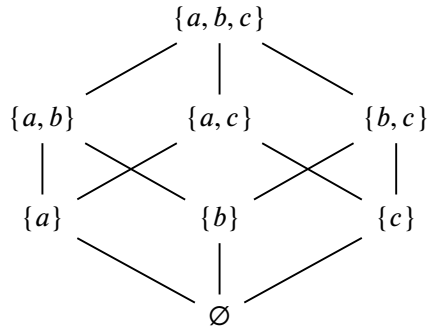



Figure 5.9.: Power set as a category.

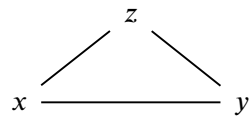



Figure 5.10.: Example of a preorder.

of the relation. We will only refer to posets for two reasons:

1. The exposition is smoother.
2. Given a preorder, computation will always involve passing to the poset representation.

This means that, given a preorder, we can consider the poset of its isomorphism classes, by means of the following equivalence relation:

$$x \simeq y \quad \equiv \quad (x \preceq y) \wedge (y \preceq x). \tag{5.1}$$

## 5.2. A poset as a category

A single poset $\langle P, \preceq \rangle$ can be described as a category, in which each point $x \in P$ is an object, and there is a morphism between $x$ and $y$ if and only if $x \preceq y$. This is a "thin" category, wich means that there is at most one morphism between two objects: For any $x, y \in P$, there exist only one relation $x \preceq y$ in $P$ (Definition 5.1). The identity morphism is given by the reflexivity property of posets, i.e. for any $x \in P$, we have $x \preceq x$. Furthermore, composition is given by the transitivity property of posets, i.e. for $x, y, z \in P$, $x \preceq y$ and $y \preceq z$ implies $x \preceq z$.

**Example 5.10.** Let's revisit Example 5.8, in which we had a poset $\mathcal{P}(\{a, b, c\})$ with order given by inclusion (Fig. 5.11). This is a category $\mathbf{C}$, with $\mathsf{Ob}_{\mathbf{C}} = \mathcal{P}(\{a, b, c\})$,

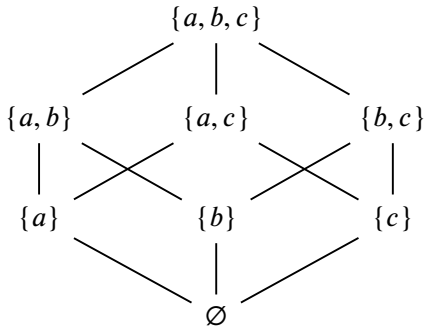



Figure 5.11.: Power set $\mathcal{P}\{a, b, c\}$ as a category.

and morphisms given by the inclusions. Note that we omit to draw self-arrows for the identity morphisms. Composition is given by the transitivity law of posets. For instance, since $\{a\} \subseteq \{a, b\}$ and $\{a, b\} \subseteq \{a, b, c\}$, we can say that $\{a\} \subseteq \{a, b, c\}$.

## 5.3. Constructing posets

We can think of the product of posets.

**Definition 5.11** (Product of posets)**.** Given two posets $\langle X, \preceq_X \rangle$ and $\langle Y, \preceq_Y \rangle$, the *product poset* is $\left\langle X \times Y, \preceq_{X \times Y} \right\rangle$, where $X \times Y$ is the Cartesian product of two sets (Definition 6.1) and the order $\preceq_{X \times Y}$ is given by:

$$\langle x_1, y_1 \rangle \preceq_{A \times B} \langle x_2, y_2 \rangle \quad \Leftrightarrow \quad (x_1 \preceq_X x_2) \wedge (y_1 \preceq_Y y_2). \tag{5.2}$$

Recalling the pizza recipes example, we have the two posets representing time and money. Given that we want to minimize both time and costs, by considering the money poset containing elements 1 USD, 2 USD, and 3 USD, and the time poset containing elements 1 h, and 2 h, one can represent the product as in Fig. 5.12.

**Example 5.12.** Consider now the two posets given in Fig. 5.13. Their product is depicted in Fig. 5.14.

### 5.3.1. Disjoint union of posets

Similarly to what we have done for sets in Section 6.1, we can think of alternatives in the poset case through their disjoint union.

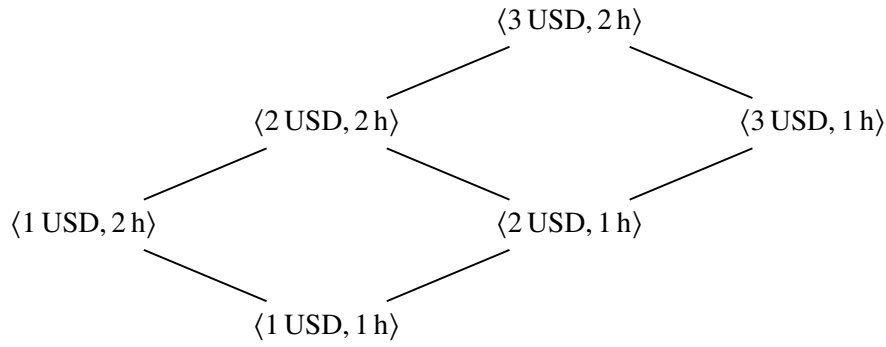



Figure 5.12.: Product poset of time and cost for pizza recipes.

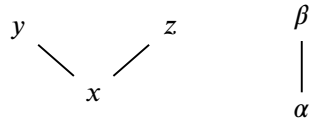



Figure 5.13.: Two posets.

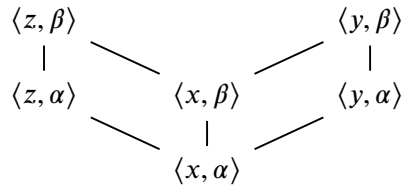



Figure 5.14.: Product of two posets.

**Definition 5.13** (Disjoint union of posets)**.** Given two posets $\langle X, \preceq_X \rangle$ and $\langle Y, \preceq_Y \rangle$, we can define their *disjoint union* $\left\langle X + Y, \preceq_{X+Y} \right\rangle$, where $X + Y$ is the disjoint union of the sets $X$ and $Y$ (Definition 6.12), and the order $\preceq_{X+Y}$ is given by:

$$x \preceq_{X+X} y \quad \equiv \quad \begin{cases} x \preceq_A y, & x, y \in X, \\ x \preceq_B y, & x, y \in Y. \end{cases} \tag{5.3}$$

**Example 5.14.** Consider the posets $X = \langle \dagger, \star \rangle$ with $\dagger \preceq_X \star$, and $Y = \langle *, \diamond, \star \rangle$, with $* \preceq_Y \diamond$ and $\diamond \preceq_Y \star$. Their disjoint union can be represented as in Fig. 5.15.

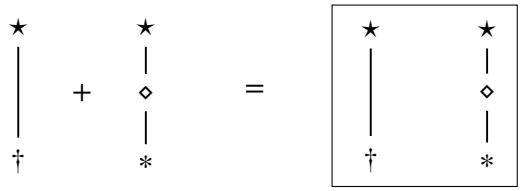


Figure 5.15.: Disjoint union of posets.

## 5.4. Staring at Pareto fronts

### 5.4.1. Chains and Antichains

**Definition 5.15** (Chain in a poset)**.** Given a poset $S$, a *chain* is a sequence of elements $s_i$ in $S$ where two successive elements are comparable, i.e.:

$$i \preceq j \Rightarrow s_i \preceq s_j. \tag{5.4}$$

**Definition 5.16** (Antichain in a poset)**.** An *antichain* is a subset $S$ of a poset where no elements are comparable. If $a, b \in S$, then $a \preceq b$ implies $a = b$.

> *Remark* 5.17. We denote the set of antichains of a poset $P$ by $\mathcal{A}P$.

> *Remark* 5.18. Note that given a poset $\langle P, \preceq \rangle$, $\emptyset \in \mathcal{A}P$ since $\emptyset$ contains no elements (and hence no comparable elements).

In the context of pizza recipes, consider the diagram reported in Fig. 5.16. The blue points represent an antichain of recipes $\{\langle 1\,\text{USD}, 2\,\text{h}\rangle, \langle 2\,\text{USD}, 1\,\text{h}\rangle\}$, i.e. recipes which do not dominate each other (one is cheaper but takes longer and the other is more expensive but quicker). The red point represents a recipe which cannot be part of the antichain, since it is dominated by $\langle 2\,\text{USD}, 1\,\text{h}\rangle$.

**Example 5.19.** Let's consider the poset $\langle P, \preceq \rangle$ where $a \preceq b$ if $a$ is a divisor of $b$ and $P = \{1, 5, 10, 11, 13, 15\}$. A chain of $P$ is $\{1, 5, 10, 15\}$. An antichain of $P$ is $\{10, 11, 13\}$.

**Example 5.20.** Consider Example 5.8. Examples of chains are

$$\{\emptyset, \{a\}, \{a, b\}, \{a, b, c\}\}, \quad \{\emptyset, \{b\}, \{b, c\}, \{a, b, c\}\}. \tag{5.5}$$

Examples of antichains are

$$\{\{a\}, \{b\}, \{c\}\}, \quad \{\{a, b\}, \{a, c\}, \{b, c\}\}. \tag{5.6}$$

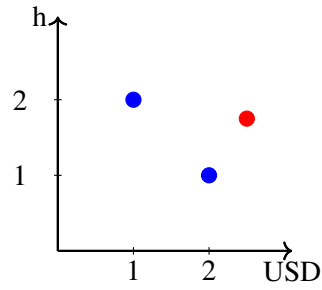



Figure 5.16.: Example of an antichain of pizza recipes.

**Example 5.21.** Suppose you have to choose a battery model based on its cost and its weight, both to be minimized. There may be models which dominate others. For instance, a model $\langle 10\,\text{USD}, 1\,\text{kg}\rangle$ is always better than a model $\langle 11\,\text{USD}, 1.1\,\text{kg}\rangle$. Also, there may be models which are incomparable, i.e. which form an antichain. For example, you cannot say whether $\langle 10\,\text{USD}, 1\,\text{kg}\rangle$ is better than $\langle 5\,\text{USD}, 2\,\text{kg}\rangle$. The incomparable models form an antichain.

### 5.4.2. Upper and lower sets

**Definition 5.22** (Upper set)**.** An *upper set* is a subset $U$ of a poset $P$ such that, if an element is inside, all elements above it are inside as well. In formulas:

$$U \text{ is an upperset} \equiv \forall x \in U, \forall y \in P : \; x \preceq y \Rightarrow y \in U. \tag{5.7}$$

*Remark* 5.23. We call $\mathsf{U}P$ the set of upper sets of $P$.

**Definition 5.24** (Lower set)**.** A *lower set* is a subset $L$ of a poset $P$ if, if a point is inside, all points below it are inside as well. In formulas:

$$L \text{ is a lower set} \equiv \forall x \in L, \forall y \in P : \; y \preceq x \Rightarrow y \in L. \tag{5.8}$$

*Remark* 5.25. We call $\mathsf{L}P$ the set of lower sets of $P$.

*Remark* 5.26. Note that if $A$ is an antichain of a poset $P$, then the set

$$I(A) = \{x : \; x \preceq y, y \in A\} \tag{5.9}$$

> is a lower set of $P$.

Consider the blue poset of pizza recipes from before. The upper and lower sets of this poset can be represented as in Fig. 5.17. The upper set can be interpreted as all the potential pizza recipes for which we can find better alternatives in the poset. Similarly, the lower set can be interpreted as all the potential pizza recipes which would be better than the ones in the poset.

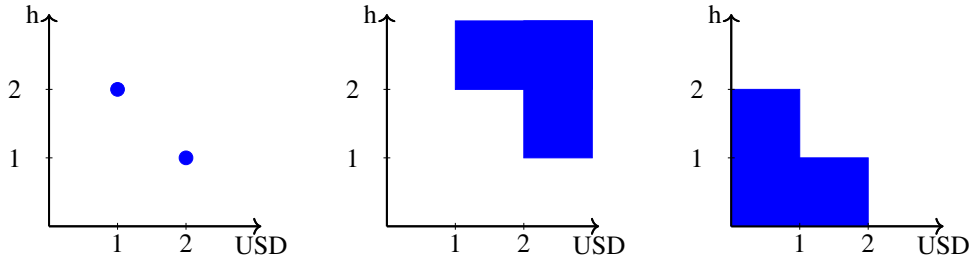



Figure 5.17.: Example of upper and lower sets of a poset of pizza recipes.

**Example 5.27** (Upper and lower sets in `Bool`)**.** The booleans $\{\texttt{F}, \texttt{T}\}$ form a poset with $\texttt{F} \leq \texttt{T}$: $(\texttt{Bool}, \leq)$ . The subset $\{\texttt{F}\} \subseteq \texttt{Bool}$ is not an upper set, since $\texttt{F} \leq \texttt{T}$ and $\texttt{T} \notin \{\texttt{F}\}$.

# 6. Combining

## 6.1. Products

We'll start off by recalling a familiar way of combining two sets, $X$ and $Y$.

**Definition 6.1** (Cartesian product of sets)**.** Given two sets $X, Y$, their *cartesian product* is denoted $X \times Y$ and defined as

$$X \times Y = \{\langle x, y\rangle \mid x \in X \text{ and } y \in Y\}. \tag{6.1}$$

**Example 6.2.** Consider the sets $X = \{1, 2, 3, 4\}$ and $Y = \{*, \dagger\}$. We have

$$X \times Y = \{\langle 1, \dagger\rangle, \langle 2, \dagger\rangle, \langle 3, \dagger\rangle, \langle 4, \dagger\rangle, \langle 1, *\rangle, \langle 2, *\rangle, \langle 3, *\rangle, \langle 4, *\rangle\}.$$

We can, however, also represent $X \times Y$ in a way which highlights its structure more:

| | | | | |
|---|---|---|---|---|
| $Y$ | $\langle 1, *\rangle$ | $\langle 2, *\rangle$ | $\langle 3, *\rangle$ | $\langle 4, *\rangle$ |
| | $\langle 1, \dagger\rangle$ | $\langle 2, \dagger\rangle$ | $\langle 3, \dagger\rangle$ | $\langle 4, \dagger\rangle$ |
| | | $X$ | | |

In particular, the cartesian product comes naturally equipped with two projection maps $\pi_1$ and $\pi_2$ which map an element of $X \times Y$ to its first and second coordinate, respectively:

$$\pi_1(\langle x, y\rangle) = x \quad \text{and} \quad \pi_2(\langle x, y\rangle) = y.$$

We will often depict the situation like this:

$$X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$$

$$x \longmapsfrom \langle x, y\rangle \longmapsto y$$

In this section, we will introduce the "categorical product". This notion generalizes the definition of a cartesian product of sets. We will see that the cartesian product exemplifies the categorical product when we are working within the category **Set**, but that in other other categories, the categorical product can show itself quite differently!

To give an introduction to the ideas, we'll work through an example which involves the familiar cartesian product of sets, but we'll view this example through eyeglasses which will highlight the "categorical product" essence of the cartesian product.

Our introductory example is as follows. Suppose you are at an engineering conference in Switzerland, and there will be a hike as a group outing. The organizers have prepared snacks to go. Each participant can choose a food from $X = \{a, b, c\}$ (think: apple, banana, carrot) and a drink from $Y = \{w, t\}$ (think: water, tea). Let $T$ denote the set of participants. The choice of snacks could be organized as depicted in Fig. 6.1, i.e., each participant chooses a food, and chooses a drink. This can be described via functions $f : T \to X$ and $g : T \to Y$.

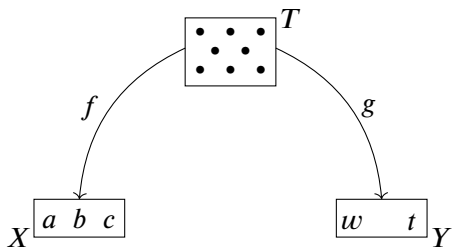



Figure 6.1.: Each participant chooses a food and a drink.

Alternatively, snacks could be pre-packaged in such a way as to allow all possible combinations of food and drink choices. This corresponds to $X \times Y$. Then the choice participants make of which lunch package they'd like is described by a single function $\phi : T \to X \times Y$, see Fig. 6.2).

Intuitively, the two situations (two choices separately, or one choice of a pre-packaged snack) are "the same" in a certain sense. In our models, we can make this precise. Specifically, if we start with the functions $f$ and $g$, we can use them to build the following function:

$$\begin{aligned} \phi_{f,g} : T &\to X \times Y \\ s &\mapsto \langle f(s), g(s) \rangle . \end{aligned}$$

Furthermore, given $\phi_{f,g}$, one can recover $f$ and $g$:

$$f = \phi_{f,g} \fatsemi \pi_1 \quad \text{and} \quad g = \phi_{f,g} \fatsemi \pi_2.$$

These two equations say that the diagram in Fig. 6.3 is commutative. The whole situation can be summarized thus: given a set $T$ and functions $f : T \to X$ and

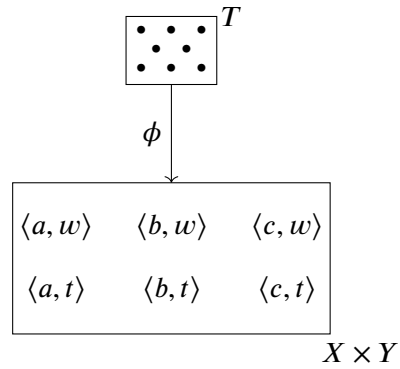



Figure 6.2.: Each participant chooses a combination of food and a drink.

$g : T \to Y$ as in Fig. 6.2, there is a unique function $\phi_{f,g} : T \to X \times Y$ such that the diagram Fig. 6.3 commutes. This is the general pattern for the definition of the categorical product, which we state now. It is probably helpful to read the definition together with the clarifying remarks that follow it.

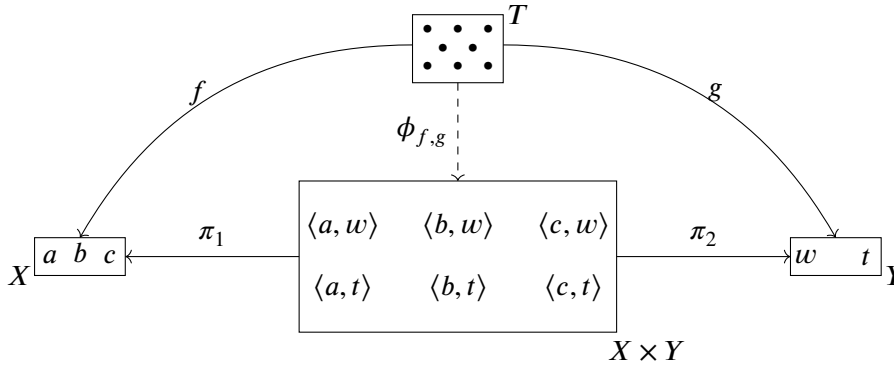



Figure 6.3.: Choosing food and drink separately is essentially the same as choosing a combination of the two.

**Definition 6.3** (Categorical Product)**.** Let $\mathbf{C}$ be a category and let $X, Y \in \mathsf{Ob}_{\mathbf{C}}$ be objects. The *product* of $X$ and $Y$ is defined by the following consituent data, satisfying the following condition.

Data:

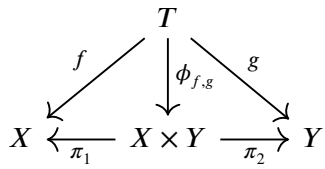
1. an object $Z \in \mathsf{Ob}_{\mathbf{C}}$ (this is "the product" of $X$ and $Y$);

2. *projection morphisms* $\pi_1 : Z \to X$ and $\pi_2 : Z \to Y$,

Condition:

1. For any $T \in \mathsf{Ob}_{\mathbf{C}}$ and any morphisms $f : T \to X, g : T \to Y$, there exists a *unique* morphism $\phi_{f,g} : T \to Z$ such that $f = (\phi_{f,g}) \fatsemi \pi_1$ and $g = (\phi_{f,g}) \fatsemi \pi_2$.

*Remark* 6.4. Diagrammatically, the condition above states that the diagrams of this form commute:




(6.2)

*Remark* 6.5. In the above definition, technically both $Z$ *and* the projection morphisms constitute the data of "the product of $X$ and $Y$". However, for simplicity, we usually refer only to $Z$ as "the product". Furthermore, we will usually use the notation $X \times Y$ to denote the product of $X$ and $Y$, in place of $Z$. Similarly, we will usually write $f \times g$ in place of $\phi_{f,g}$. The reason we do not do this directly in the definition itself is the following. In general, for fixed $X$ and $Y$, there may be several different objects $Z$ (together with projection morphisms) that satisfy the definition of being "the product of $X$ and $Y$". Thus, there is, technically, no such thing as "*the*" (unique) product of $X$ and $Y$. However, one can prove that any two candidates which satisfy the definition of being "the product of $X$ and $Y$" will necessarily be isomorphic in a canonical manner. Thus, for simplicity, we will sometimes be slightly sloppy and speak of "the product of $X$ and $Y$" as if it were unique. In many categories there is also indeed a choice for "the product of $X$ and $Y$" that we are used to. For example, in the category **Set**, given sets $X$ and $Y$, the familiar choice for "the product of $X$ and $Y$" is the cartesian product $X \times Y$". However, other representatives of the product of $X$ and $Y$ are possible! Example 6.7 illustrates this.

*Remark* 6.6. The condition in the definition of the categorical product is know as the "universal property of the product". We will attempt to explain this naming. The stated condition involves the product $Z$ of $X$ and $Y$ *interacting* with every possible choice of object $T$ and every possible choice of morphisms $f : T \to X$ and $g : T \to Y$. We think of the ambient category $\mathbf{C}$ as "the universe" (or the "context"), and this condition states how the product must interact "with the

whole universe". We choose the letter "$T$" because we think of this as a "test object" (similar e.g. to how, in electrodynamics, a "test charge" is used to probe an electromagnetic field).

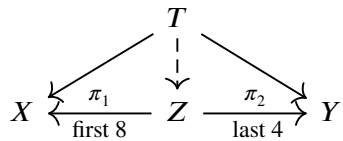
**Example 6.7.** Suppose that as a manufacturer, you want to label your products with

- A production date (8-digit code), and
- a model number (4-digit code).

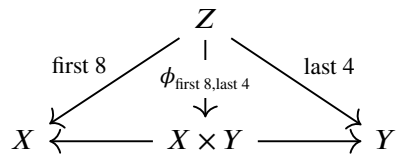
Instead of two separate labels, you can make one:

$$\underbrace{20210115}_{\text{date}}\,\underbrace{5900}_{\text{model}}$$

We call this single label the *product code*. Let $Z$ denote the set of all product codes, and consider the maps $\pi_1 : Z \to X$, and $\pi_2 : Z \to Y$ which, respectively, map a 12-digit product code to its first 8 digits and its last 4 digits. One may check that $Z$, together with the map $\pi_1$ and $\pi_2$, will satisfy the definition of "the product of $X$ and $Y$".




However, $Z$ is not precisely the cartesian product of $X$ and $Y$ (which we will call $X \times Y$). The elements of $Z$ are 12-digit codes, while elements of $X \times Y$ are pairs $\langle x, y \rangle$ where $x$ is a 8-digit code and $y$ is a 4-digit code. Since both $Z$ and $X \times Y$ satisfy the definition of categorical product, they must, by Remark 6.5, be isomorphic. To see concretely what this isomorphism between them looks like, note that there is a unique map $\phi_{\text{first 8,last 4}}$ making the following diagram commute:




Concretely, $\phi_{\text{first 8,last 4}} : Z \to X \times Y$ maps for instance

$$202101155900 \mapsto \langle 20210115, 5900 \rangle .$$

One can readily show that $\phi_{\text{first 8,last 4}}$ is an isomorphism.

Now we will take a small tour to see some examples of how the categorical product may look in categories different than the category **Set** of sets and functions.

**Example 6.8.** Let $m, n \in \mathbb{N}$, and draw an arrow $m \to n$ if $m$ divides $n$. For instance, 6 divides 12 and hence there is an arrow $6 \to 12$. The product between any two $m, n \in \mathbb{N}$ in this category is given by the greatest common divisor.

**Example 6.9.** Let's consider the ordered set $\langle \mathbb{R}, \leq \rangle$, where given $x_1, x_2 \in \mathbb{R}$ we can draw an arrow $x_1 \to x_2$ if $x_1 \leq x_2$. By following the products's commutative diagram, we know that the product of $x_1$ and $x_2$ is a $z \in \mathbb{R}$ such that

- $z \leq x_1$;
- $z \leq x_2$;
- For all $x \in \mathbb{R}$ with $x \leq x_1$ and $x \leq x_2$, we have $x \leq z$.

In other words, the product of $x_1, x_2 \in \mathbb{R}$ is given by $\min\{x_1, x_2\}$, and is also called *meet*.

**Example 6.10.** Let $S$ be a set, and $X, Y \subseteq S$ subsets. We can draw an arrow $X \to Y$ if $X \subseteq Y$. By following the product's commutative diagram, it is easy to see that the product of $X$ and $Y$ is given by $X \cap Y$.

**Example 6.11.** Suppose that we are designing a vehicle, and we are thinking about choices of engine. Both electric engines and internal combustion engins can produce motion, but each from a different source of energy. The electric engine uses electric energy; the internal combustion engine uses gasoline. The situation is depicted in Fig. 6.4, using the interpretation of the arrows that we have introduced for engineering design components. Namely, the arrow from motion to gasoline represents the internal combustion engine, and its direction is to be read as follows: given the desired functionality motion, internal combustion engine provides a way of getting it using gasoline. The other arrow in the figure represents the component electric engine, and is interpreted in a similar way.

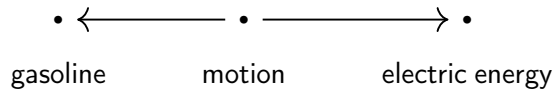



Figure 6.4.: Alternative ways to generate motion.

We could also consider building a hybrid vehicle, where we can obtain motion from **either** gasoline **or** electric energy (Fig. 6.5).

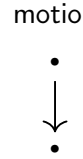



Figure 6.5.: We can generate motion from either gasoline or electric energy.

## 6.2. Coproduct

There exists a "dual" notion to "product" that is called "coproduct". Just like the notion of categorical product generalized the definition of the cartesian product of two sets, the categorical coproduct generalizes the definition of the *disjoint union* of two sets.

Recall that given sets $X$ and $Y$, their disjoint union $X + Y$ is a set that contains a distinct copy of $X$ and $Y$ each. If an element is contained in both $X$ and $Y$, then there will be two distinct copies of it in the disjoint union $X + Y$.

**Definition 6.12** (Disjoint union of sets)**.** The *disjoint union* (or sum) of two sets $X$ and $Y$ is

$$X + Y = \{\langle 1, x\rangle \mid x \in X\} \cup \{\langle 2, y\rangle \mid y \in Y\}. \tag{6.3}$$

**Example 6.13.** Consider the sets $\{\star, \diamond\}$ and $\{*, \dagger\}$. Their disjoint union can be represented as in Fig. 6.6.

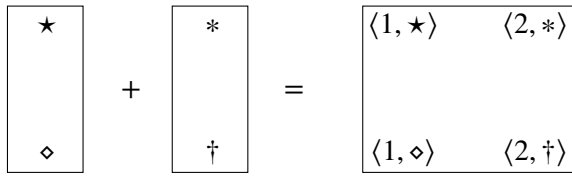



Figure 6.6.: Example of a disjoint union of sets.

We can define the disjoint union of a set with itself; this corresponds to having two distinct copies of the set (Fig. 6.7).

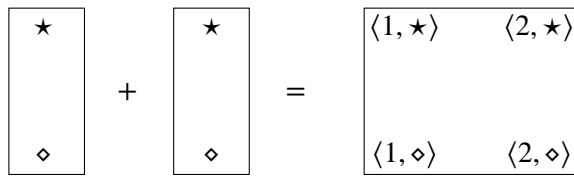



Figure 6.7.: Disjoint union of a set with itself .

As mentioned above, the disjoint union is a particular instance – in the category **Set**– of the notion of "coproduct". We will now give the definition of a coproduct in an arbitrary category. Note that it is very similar to the definition that we gave, in the previous section, for the product – but with a few twists. Analogous remarks to those we gave following the definition of the product apply here!

**Definition 6.14** (Coproduct)**.** Let $\mathbf{C}$ be a category and let $X, Y \in \mathsf{Ob}_{\mathbf{C}}$ be objects. The *coproduct* of $X$ and $Y$ consists of the following constituent data, satisfying the following condition.

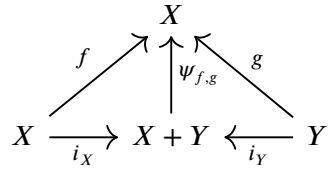
Data:

1. an object $Z \in \mathsf{Ob}_{\mathbf{C}}$ ("the coproduct" of $X$ and $Y$)
2. *injection morphisms* $\iota_1 : X \to Z$ and $\iota_2 : Y \to Z$

Condition:

1. For any $T \in \mathsf{Ob}_{\mathbf{C}}$ and any morphisms $f : X \to T, g : Y \to T$, there exists a *unique* morphism $\psi_{f,g} : Z \to T$ such that $f = \iota_1 \fatsemi \psi_{f,g}$ and $g = \iota_2 \fatsemi \psi_{f,g}$.

*Remark* 6.15. Diagrammatically, the condition above states that diagrams of this form commute:




(6.4)

*Remark* 6.16. Similarly as was the case with the categorical product, "the coproduct" of $X$ and $Y$ is unique only "up to isomorphism". Nevertheless, we will usually simply write $X + Y$ for "the" coproduct (in place of $Z$ above), and we will usually write $f + g$ in place of $\psi_{f,g}$.

**Example 6.17.** Let's consider two battery producers, each producing specific battery technologies. The first producer is able to produce a set $A = \{\mathsf{LiPo}, \mathsf{LCO}, \mathsf{NiH2}\}$ of technologies, and the second one a set $B = \{\mathsf{LFP}, \mathsf{LMO}, \mathsf{LiPo}\}$. Each technology has a specific price, belonging to a set of prices $P = \{50\,\mathrm{USD}, 60\,\mathrm{USD}, 70\,\mathrm{USD}, 80\,\mathrm{USD}\}$. We specify the price mappings for different technologies via the functions $f : A \to P$ and $g : B \to P$. A battery vendor wants to sell batteries from both producers and wants to create a battery catalogue, which needs to take into account which technology comes from which producer, to be able to distribute the earnings from the sales fairly. To this end, the disjoint union of the sets of technology is considered:

$$A + B = \{\langle 1, \mathsf{LiPo}\rangle, \langle 1, \mathsf{LCO}\rangle, \langle 1, \mathsf{NiH2}\rangle, \langle 2, \mathsf{LFP}\rangle, \langle 2, \mathsf{LMO}\rangle, \langle 2, \mathsf{LiPo}\rangle\}.$$

It is possible to map each technology in $A, B$ to its own representative in $A + B$ via

the so-called injection maps:

$$\begin{aligned} \iota_A \colon A &\to A + B \\ a &\mapsto \langle 1, a \rangle, \\ \iota_B \colon B &\to A + B \\ b &\mapsto \langle 2, a \rangle. \end{aligned}$$

This situation is graphically represented in Fig. 6.8, and mimics the coproduct diagram presented in Definition 6.14.

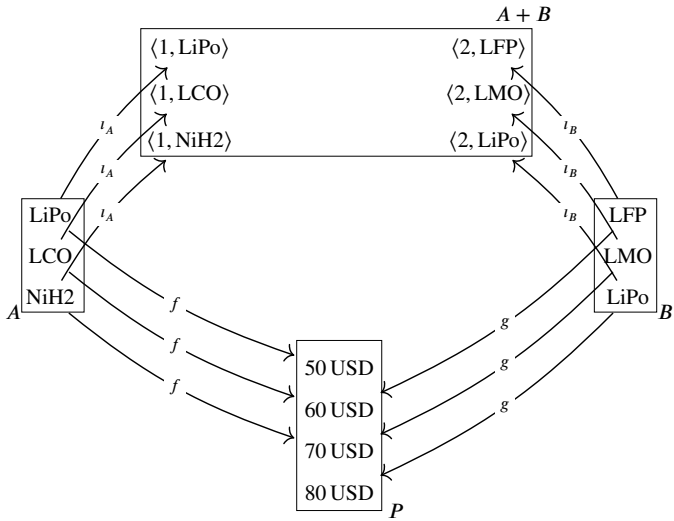



Figure 6.8.: Battery technologies, companies, prices, and a catalogue.

Here, the universal property says that there is a **unique** function $f + g \colon A + B \to P$ such that

$$\iota_A \fatsemi (f + g) = f \text{ and } \iota_B \fatsemi (f + g) = g.$$

If we take a $x \in A + B$ is either "from $A$ or from $B$":

$$\text{either } \exists a \in A \colon x = \iota_A(a) \text{ or } \exists b \in B \colon x = \iota_B(b).$$

From this, we can deduce that the desired map $f + g$ is:

$$\begin{aligned} f + g \colon A + B &\to P \\ x &\mapsto \begin{cases} f(x), & \text{if } x = \iota_A(a), \quad a \in A, \\ g(x), & \text{if } x = \iota_B(b), \quad b \in B. \end{cases} \end{aligned}$$

This is a specific example of **Set/FinSet**, in which the coproduct is a generalization of the concept of disjoint union. Now, we could spontaneously ask ourselves: why does

the union not "suffice" for the coproduct definition in **Set**? To see this, let's consider the same situation as before, but now having the catalogue of technologies given by $A \cup B$ (Fig. 6.9). The interpretation of maps $f, g$ does not change, and injections work as depicted. Note, however, that when asked for a map from the technology $\mathsf{LiPo} \in A \cup B$, we have no notion of the company which produces it, and we are therefore unsure whether to assign it to $f(\mathsf{LiPo}) = 50\,\text{USD}$ or $g(\mathsf{LiPo}) = 80\,\text{USD}$. Indeed, the unique map $f + g$ required by the universal property of the coproduct cannot exist, since in case $A \cap B \neq \emptyset$, any element $x \in A \cap B$ should be simultaneously sent to $f(x)$ and $g(x)$.

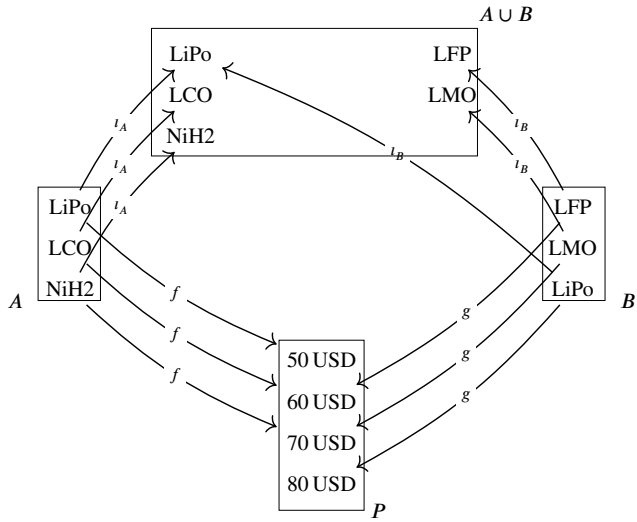



Figure 6.9.: Example of why the union is not the coproduct in **Set**.

**Example 6.18.** Given $X, Y \in \mathsf{Ob}_{\mathbf{Rel}}$ (so $X$ and $Y$ are sets) their coproduct is the disjoint union $X + Y$. The disjoint union of sets comes equipped with inclusion functions $\iota_X : X \to X + Y$ and $\iota_Y : Y \to X + Y$. If we turn these functions into relations

$$R_{\iota_X} \subseteq X \times (X + Y)$$
$$R_{\iota_Y} \subseteq Y \times (X + Y).$$

then these are the injection morphisms for the coproduct in **Rel**. As an aside, we note that in **Rel** products and coproducts are *both* given by the disjoint union of sets. We will see later why this is might be expected.

**Example 6.19.** Let $m, n \in \mathbb{N}$, and draw an arrow $m \to n$ if $m$ divides $n$. For instance, 6 divides 12 and hence there is an arrow $6 \to 12$. The coproduct between any two $m, n \in \mathbb{N}$ in this category is given by the least common multiple.

**Example 6.20.** Let's consider the ordered set $\langle \mathbb{R}, \leq \rangle$, where given $x_1, x_2 \in \mathbb{R}$ we can draw an arrow $x_1 \to x_2$ if $x_1 \leq x_2$. By following the coproduct's commutative diagram, we know that the coproduct of $x_1$ and $x_2$ is a $z \in \mathbb{R}$ such that

- $x_1 \leq z$;
- $x_2 \leq z$;
- For all $x \in \mathbb{R}$ with $x_1 \leq x$ and $x_2 \leq x$, we have $z \leq x$.

In other words, the coproduct of $x_1, x_2 \in \mathbb{R}$ is given by $\max\{x_1, x_2\}$, and is also called *join*.

**Example 6.21.** Let $S$ be a set, and $X, Y \subseteq S$ subsets. We can draw an arrow $X \to Y$ if $X \subseteq Y$. By following the coproduct's commutative diagram, it is easy to see that the coproduct of $X$ and $Y$ is given by $X \cup Y$, i.e., the "smallest" set containing both $X$ and $Y$.

**Example 6.22.** We can define a category **Vect** in, composed of:

- *Objects*: vector spaces;
- *Morphisms*: linear maps;
- *Identity morphisms*: identity maps;
- *Composition*: composition of linear maps.

It is a good exercise to prove that **Vect** really forms a category. In the following, we want to look at the coproduct in **Vect**. It is given by the *direct sum* of vector spaces. Recall that given vector spaces $V$ and $W$, their direct sum is the set

$$V \oplus W := \{\langle v, w \rangle \mid v \in V, w \in W\},$$

equipped with a notion of addition and scalar multiplication derived component-wise from $V$ and $W$. For addition, this means that given vectors $\langle v_1, w_1 \rangle, \langle v_2, w_2 \rangle \in V \oplus W$, their sum in $V \oplus W$ is

$$\langle v_1, w_1 \rangle + \langle v_2, w_2 \rangle := \langle v_1 + v_2, w_1 + w_2 \rangle.$$

The injection morphisms for the coproduct are given by:

$$\begin{aligned} \iota_V : V &\to V \oplus W \\ v &\mapsto \langle v, 0_W \rangle, \\ \iota_W : W &\to V \oplus W \\ w &\mapsto \langle 0_V, w \rangle, \end{aligned}$$

where $0_V$ and $0_W$ represent the zero vectors in $V$ and $W$. Let's now look at the universal property in this case, by considering any vector space $U \in \mathsf{Ob}_{\mathbf{Vect}}$, and linear maps $S : V \to U, T : W \to U$. The universal property says that we need a unique

linear map $S + T : V \oplus W \to U$ such that $S = \iota_V \fatsemi h$ and $T = \iota_W \fatsemi g$. By taking any $\langle v, w \rangle \in V \oplus W$, we can write:

$$\begin{aligned} h(\langle v, w \rangle) &= h(\langle v, 0_W \rangle + \langle 0_V, w \rangle) \\ &= h(\iota_V(v) + \iota_W(w)) \\ &= h(\iota_V(v)) + h(\iota_W(v)) && (h \text{ is linear}) \\ &= (\iota_V \fatsemi h)(v) + (\iota_W \fatsemi h)(w) \\ &\overset{!}{=} Sv + Tw. \end{aligned}$$

We can hence write the map $S + T$ as

$$\begin{aligned} S + T : V \oplus W &\to U \\ \langle v, w \rangle &\mapsto Sv + Tw. \end{aligned}$$

**Example 6.23** (Adapted from [**spivak2014category**])**.** We can define a category of graphs **Grph**. Objects of this category are graphs $G = \langle V, A, s, t \rangle$, composed of:

- A set of *vertices* $V$;
- A set of *arrows* $A$;
- A *source* function $s : A \to V$, mapping each arrow to its source vertex;
- A *target* function $t : A \to V$, mapping each arrow to its target vertex.

Morphisms are *graph homomorphisms*. Given graphs $G = \langle V, A, s, t \rangle$ and $G' = \langle V', A', s', t' \rangle$, a graph homomorphism $f : G \to G'$ is given by maps $f_0 : V \to V'$ and $f_1 : A \to A'$, such that the following diagrams commute:

$$\begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ \downarrow{\scriptstyle s} & & \downarrow{\scriptstyle s'} \\ V & \xrightarrow{f_0} & V' \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ \downarrow{\scriptstyle t} & & \downarrow{\scriptstyle t'} \\ V & \xrightarrow{f_0} & V' \end{array}$$

Intuitively, all this is saying is that "arrows are bound to their vertices", meaning that if a vertex $v_1$ is connected to $v_2$ via an arrow $a$, the vertices $f_0(v_1)$ and $f_0(v_2)$ have to be connected via an arrow $f_1(a)$. We are now ready to define the coproduct in **Grph**. Given two graphs $G = \langle V, A, s, t \rangle$ and $G' = \langle V', A', s', t' \rangle$, their coproduct is a graph

$$G + G' = \left\langle V + V', A + A', s + s', t + t' \right\rangle.$$

In $G + G'$, an arrow connects $v_1$ to $v_2$ if:

- $v_1, v_2 \in V$ or $v_1, v_2 \in V'$, i.e., if both vertices belong to the same graph, and
- an arrow between from $v_1$ to $v_2$ exists in $G$ or $G'$.

Given $s : A \to V$ and $s' : A' \to V'$, we have:

$$\begin{aligned} s + s' : A + A' &\to V + V' \\ x &\mapsto \begin{cases} s(x) & \text{if } x \in A \\ s'(x) & \text{if } x \in A'. \end{cases} \end{aligned}$$

and

$$t + t' : A + A' \to V + V'$$
$$x \mapsto \begin{cases} t(x) & \text{if } x \in A \\ t'(x) & \text{if } x \in A'. \end{cases}$$

This is nicely graphically representable. Consider two graphs as in Fig. 6.10.

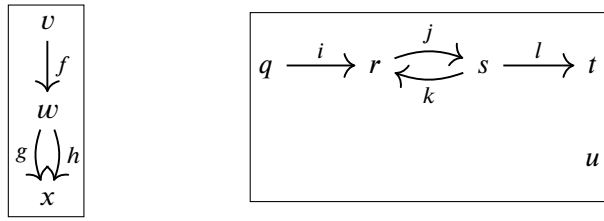



Figure 6.10.: Example of graphs for which we want to consider the coproduct.

Their coproduct is a graph including the "disjoint union" of both original graphs, without connecting them (Fig. 6.11).

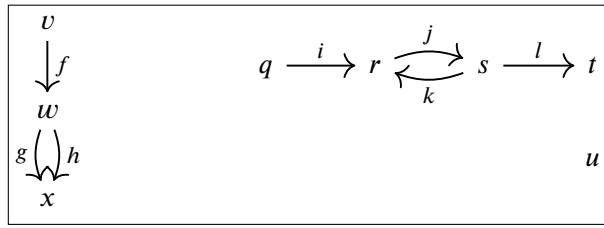



Figure 6.11.: Example of coproduct of graphs.