0. Preliminaries

A *set* X is a collection of *elements*. When x is an element of X, we write $x \in X$. When a set X consists of specified elements $x, y, \dots$, we write $X = \{x, y, \dots\}$. What follows is a list of sets whose notation and name we fix throughout the sequel.

- the *empty* or *null* set $\varnothing = \{\}$
- the generic *singleton* $\star = \{*\}$
- the *Booleans* $\mathbb{B} = \{\top, \bot\}$, where $\top$ is "true" and $\bot$ is "false"
- the generic n-element set $\mathbf{n} = \{0, 1, \dots n-1\}$
- the *natural numbers* $\mathbb{N} = \{0, 1, \dots\}$
- the *integers* $\mathbb{Z} = \{\cdots -1, 0, 1, \dots\}$

Two sets A, B are said to be equal, in which case we may write $A = B$, when $x \in A$ if and only if $x \in B$. We note that $\varnothing = \mathbf{0}$, and hence these can be used interchangeably. The *cardinality* of a set is its number of elements. Although $\star$ and $\mathbf{1}$ are not equal, they both aspire towards being "generic" sets of cardinality 1 and are hence interchangeable since their respective elements are mere placeholders. In both cases, we will choose the notation whose evocation best matches the context.

We say A is a *subset* of X and write $A \subseteq X$ when $x \in A$ implies $x \in X$. When we wish to define a subset B from a set X that keeps only the elements of X that satisfy some *predicate*, i.e. *condition*, $\mathbf{P}$, we write $B = \{x \in X \mid \mathbf{P}(x)\}$. One reads each term in this expression using the following dictionary:

$\{\}$	"the set of"
$x \in X$	"x's in X"
$\mid$	"such that"
$\mathbf{P}(x)$	"the predicate $\mathbf{P}$ holds for x"

Alternatively, we may also write this set via the shorthand $X|_{\mathbf{P}}$, which should be read as "the set X, *restricted* to its elements that satisfy $\mathbf{P}$." For example, we write $\mathbb{Z}|_{\leq 3} = \{\dots, -3, -2, \dots, 2, 3\}$, or sometimes, when unambiguous, as simply $\mathbb{Z}_{\leq 3}$, omitting the '$|$' symbol altogether. This style of specifying sets—via elements of an ambient set for which a specified predicate holds—is called *set-builder* notation. We can express many familiar constructions this way. For example, the *powerset* $\mathcal{P}X = \{S \subseteq X\}$ is the set of all subsets of X. Furtheremore, given two subsets $A, B \subseteq X$, i.e. elements $A, B \in \mathcal{P}X$, we can use this notation to define the following two new subsets.

- the *union* $A \cup B = \{x \in X \mid x \in A \text{ or } x \in B\}$
- the *intersection* $A \cap B = \{x \in X \mid x \in A \text{ and } x \in B\}$

We say that two subsets $A, B \subseteq X$ are *disjoint* when $A \cap B = \varnothing$. We say that A, B are *exhaustive* when $A \cup B = X$. We say that A, B *partition* X when they are both disjoint and exhaustive.

More generally, we can write some *expression* to the left of the '$|$' instead of just the element $x \in X$. In this case, we use the convention that the set to which each element in the expression belongs is written to the right of the '$|$' symbol. For example, we can define the set $\mathbb{Q}$ of *rational numbers* using this notation:

$$\mathbb{Q} = \{\tfrac{p}{q} \mid p, q \in \mathbb{Z}, q \neq 0\}$$

In this case our set consists of expression given by the fractions p/q, whose constituent terms p, q are subject to the condition that they are integers, and that q

is subject to the condition that it does not equal 0. Note that we have thus far employed commas as an informal "and."

The flip side of restricting a set is *excluding* from a set. More precisely, when $S \subseteq X$, we write $X \setminus S = \{x \in X \mid x \notin S\} = X|_{\notin S}$. Then, if $\mathbf{P}'$ is the negation of $\mathbf{P}$, we have that $X|_{\mathbf{P}'} = X \setminus X|_{\mathbf{P}}$. We will more often characterize sets via restrictions than exclusions.

We now introduce some fundamental abstract operations on sets. Unlike the previous operations—restrictions, unions, intersections, exclusions—these are not *internal* operations—those on a subset relative to some ambient set—but rather are *external* operations on sets themselves. Given two sets X, Y, we define their *Cartesian product*

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\}$$

as the set of *ordered pairs* whose first entry is in X and second entry in Y. Given these sets, we can also form their *disjoint union* or *sum* $X + Y$ as the set consisting of the X-elements alongside the Y-elements. This is not the same as the union! One takes the union of two *subsets* $A, B \subseteq X$ of a fixed *ambient set* X by considering the set of X-elements that are present in at least one of A or B. This means that an X-element that appears in both A and B is not double counted. However, we need this ambient set X in order to even compare elements of A with those of B, since these a priori have nothing to do with each other. In practice, it has historically been frustrating to articulate the distinction between these two concepts since most of the time we express the elements of our two sets X, Y using a shared "alphabet of symbols," which implicitly plays the role of ambient set. Generically, however, if no interaction is pre-specified between two sets X, Y, we cannot even think about what it would mean to form the classic union, since the elements of each exist beyond comparison with each other.

Returning to internal operations, we will also be interested in *quotients* of sets. This means that, if we have some notion of "equivalence" that is coarser than that of equality of elements, we can partition the set into mutually disjoint and collectively exhaustive classes of "equivalent" elements. More formally, an *equivalence relation* $\sim$ on a set X is a certain kind of predicate on $X \times X$. We use *infix* notation $x \sim x'$ in place of $\sim (x, x')$ to express that "x is related, via $\sim$, to x'." This relation must satisfy the following conditions.

- $x \sim x$ (*reflexivity*)
- $x \sim x' \Leftrightarrow x' \sim x$ (*symmetry*)
- $x \sim x', x' \sim x'' \Rightarrow x \sim x''$ (*transitivity*)

Given $\sim$ on X, we write $[x]$ for the *equivalence class* of x, i.e. the set

$$X|_{\sim x} = \{x' \in X \mid x' \sim x\}$$

of X-elements equivalent to x. We then define the *quotient* $X_{/\sim}$ as the set of $\sim$-equivalence classes of X, i.e. $X_{/\sim} = \{[x] \mid x \in X\}$. Consider the following example. Let A consist of the set of points in the United States. Define $\sim$ by $x \sim y$ if and only if x and y belong to the same state. Then, if p is some point on the Duke University campus, $[p]$ can be identified with the set of points in North Carolina. We then see that $A_{/\sim} = \{\text{the fifty states}\}$.

A function, or henceforth, *map* f of type $X \to Y$, for sets X and Y, is a rule that assigns each element $x \in X$ precisely one element $fx \in Y$. We call X the *domain* and Y the *codomain* of f. We often refer to an element of the domain as an *argument* and an element of the codomain as a *value*. We say f *takes arguments* in X and *takes values* in Y. When applying a map f to an argument x, we say it *evaluates* to the value fx. When we specify a map by both its type and its rule, we use the notation

$$f : X \to Y :: x \mapsto fx,$$

which should be read using the following dictionary

f	"the map f"
$:$	"of type"
$X \to Y$	"X to Y"
$::$	"given by"
$x \mapsto fx$	"x maps to fx"

We write $X \xrightarrow{f} Y$ as shorthand for $f : X \to Y$. Sometimes, instead of inserting '::' between the type and the rule, we simply place the rule below the type so as to align elements with their sets:

$$X \xrightarrow{f} Y$$
$$x \mapsto fx$$

Two maps $f, g : X \to Y$ are said to be equal precisely when $fx = gx$ for all arguments $x \in X$.

It may be worthwhile to name a map by its rule. We call such a specification an *anonymous declaration.* If the intricacy of a situation calls for careful formality, we will use λ-calculus syntax, which would express our above map via the notation

$$\lambda x.fx : X \to Y$$

This is particularly useful in expressing situations in which an argument maps to a function. For example, consider the map $I : \mathcal{C}^\infty \to \mathcal{C}^\infty$ that sends a smooth map f to the smooth map given by mapping x to $\int_0^x f(t)dt$. We can write this as

$$I : \mathcal{C}^\infty \to \mathcal{C}^\infty :: f \mapsto \lambda x. \int_0^x f(t)dt \tag{1}$$

More often, however, we will use the less formal notation $f(-)$, which should evaluate an argument x by substituting x in place of the $-$. For example:

$$[\int_0^x (-)dt](1) = x$$

If our map involves arithmetic, we will use $\square$ in place of $(-)$ to avoid ambiguity.

For every set X, its *identity map* is defined as

$$\mathbb{1}_X : X \to X :: x \mapsto x.$$

Given maps

$$f : X \to Y :: x \mapsto fx \text{ and } G : Y \to Z :: y \mapsto gy$$

we can form their *composition* $g \circ f$, or simply gf, as

$$gf : X \to Z :: x \mapsto gfx.$$

The composition and identity satisfy two important axioms.

- *(associativity)* $[f \circ g] \circ h = f \circ [g \circ h]$, whenever these are composable.
- *(identity)* $f \circ \mathbb{1}_X = f = \mathbb{1}_Y \circ f$, for $f : X \to Y$.

The following elementary examples serve to demonstrate that map composition and identity generalize familiar arithmetic operations and identities.

$$(\square + n) \circ (\square + m) = \square + (n + m)$$
$$(\square \cdot n) \circ (\square \cdot m) = \square \cdot (nm)$$
$$(\square + n) \circ (\square + 0) = (\square + 0) \circ (\square + n) = \square + n$$
$$(\square \cdot n) \circ (\square \cdot 1) = (\square \cdot 1) \circ (\square \cdot n) = \square \cdot n$$

We would be remiss not to point out the following frustration. When written as a string of arrows, our situation is represented by the *diagram*

$$X \xrightarrow{f} Y \xrightarrow{g} Z,$$

which automatically renders in many minds as fg instead of gf. This unfortunate convention is a consequence of the '$f(x)$' notation, due to Euler, for the evaluation of an argument x by a map f. When treating historically ossified contexts like linear algebra, we will bend to this convention. In situations for which it would heretical to do so, however, we write the above composition in *diagrammatic order* as $f \mathbin{/\!\!/} g$, to be read as "f then g." In such contexts, we will also write $x \mathbin{/\!\!/} f$ in place of $f(x)$. This is justified by interpreting each element $x \in X$ as the map

$$x : \star \to X :: * \mapsto x.$$

This notation, with equal precision yet more succinctness and or evocativeness, recaptures familiar facts in terms of map algebra. For example, we can reconceptualize "completing the square" as factoring—via $\circ$ not $\times$—a quadratic map

$$q : \mathbb{R} \to \mathbb{R} :: x \mapsto ax^2 + bx + c$$

into the map composition

$$q = L_v \circ \square^2 \circ L_h$$

for some linear polynomial maps L_v, L_h. For example, the equality on values

$$q(x) = x^2 - 6x + 16 = (x - 3)^2 + 7$$

can now be recast as the equality on maps

$$q = (\square + 7) \circ \square^2 \circ (\square - 3).$$

This has the advantage of assessing q directly without considering its evaluation on arguments. Furthermore, this modularizes q into composable sub-processes. This decomposition, although obscured by the classic notation, is, as we shall soon see, precisely why completing the square solves quadratic equations.

Sometimes it's meaningful to consider a map which takes a pair of arguments; for example addition $+$ or multiplication $\times$ of numbers. A *binary operation* $\odot$ taking pairs of X elements as argument and outputting Y values can be conceived of as a map $\odot : X \times X \to X$. We use the infix notation $x \odot y$ as shorthand for $\odot(x, y)$. A set X with a binary operation defined on it is called a *magma.* For example, addition and multiplication of pairs of integers yields an operations of type $\mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$. Suppose we wish to iterate $\odot$ on more than two arguments:

$$(((w \odot x) \odot y) \odot z).$$

This operation takes 4 inputs. More generally, we define an n-ary operation as one that takes n arguments. Such an operation has as domain the *Cartesian power*

$$X^n = \{(x_0, \ldots, x_{n-1}) \mid \forall i \in \mathbf{n}, x_i \in X\}.$$

We refer to elements of a Cartesian power as *tuples*. Given a binary operation, the number of possible n-ary operations we can form from it grows wildly with n. The situation dramatically simplifies if we suppose further that $\odot$ is *associative*:

$$x \odot (y \odot z) = (x \odot y) \odot z.$$

A set X with an associative binary operation is often called a *semigroup*. We can then simply write $x \odot y \odot z$ without concern for bracketing. Iteratively applying associativity allows for the unique extension of $\odot$ to the following n-ary operator that takes as argument any tuple $(x_j)_{j=0}^{n-1} = (x_0, \ldots, x_n) \in X^n$.

$$\bigodot_{j=0}^{n-1} : X^n \to Y :: (x_j)_{j=0}^{n-1} \mapsto x_0 \odot \cdots \odot x_{n-1}$$

Note that, by convention, we simply write $\bigodot_{j=0}^{n-1} x_i$ in place of $\bigodot_{j=0}^{n-1} (x_i)_{i=0}^{n-1}$. When $\odot$ is invariant under swapping arguments, we say it is *commutative*:

$$x \odot y = y \odot x.$$

In this situation, since the order does not matter, $\odot$ can be extended to an operation that takes any non-empty finite set J's worth of arguments. This can be encoded by the *indexed tuple* $(x_j)_{j \in J}$, where we call J the *indexing set*. We call the extension of $\odot$ to such arguments a (finitely) *indexed* operation. Since an indexed tuple $(x_j)_{j \in J}$ can be seen as a map $J \to X :: j \mapsto x_j$, the domain of this operation should be the set of such maps $[J \to X]$. We then denote the indexed operation as follows.

$$\bigodot_{j \in J} : [J \to X] \to Y.$$

Note that, since the n-ary version can be seen as the special case of the above, we have that X^n is in some sense "the same" as $[\mathbf{n} \to X]$. We will investigate this sameness in future lectures.

Sometimes, for the sake of encoding structure more systematically, we may also be interested in the *nullary* version of $\odot$, i.e. one that takes zero arguments. What exactly could this mean? The idea is that this should be the X element, if one such exists, that is inert under $\odot$.

Given two indexing sets J, I with indexed tuples $(x_j)_{j \in J}$ and $(x_j)_{j \in I}$ in X, we can form the combined indexed tuple $(x_j)_{j \in J+I}$ with indexing set $J + I$. Applying this situation to our operators yields the following identity.

$$\bigodot_{j \in J+I} x_i = \left[\bigodot_{j \in J} x_i\right] \odot \left[\bigodot_{j \in I} x_i\right]$$

Note that, in the situation that I is the empty set $\varnothing$, $J + I$ is essentially just J. This reduces the above to:

$$\bigodot_{j \in J} x_i = \left[\bigodot_{j \in J} x_i\right] \odot \left[\bigodot_{j \in \varnothing} x_i\right]$$

When J is a singleton $\{j\}$, and we denote x_j simply as x, we have the following.

$$x = x \odot \left[\bigodot_{j \in \varnothing} x_i\right]$$

Since $J + \varnothing$ and $\varnothing + J$ are both equivalent to J, this is also equal to swapping factors on the right hand side. Therefore we *define* the nullary $\odot$ operation to be the constant map that picks out the inert or *unit* element ϵ of $\odot$. More formally, we say that $\odot$ satisfies *unitality* with *unit* ϵ when for any $x \in X$:

$$\epsilon \odot x = x = x \odot \epsilon.$$

A set X with an associative operation that has a unit is called a *monoid*. Let's apply this principle to the operations $+$ and $\times$, whose n-ary extensions are denoted with the symbols $\sum$ and $\prod$, respectively.

$$\sum_{j\in\varnothing} x_j = 0 \qquad \prod_{j\in\varnothing} x_j = 1.$$

In the special case that $x_j = j$, the latter reduces to the fact that $0! = 1$.

We may also consider the union and intersection as binary operations $\mathcal{P}X \times \mathcal{P}X \to \mathcal{P}X$. These are associative and unital with the following nullary forms:

$$\bigcup_{j\in\varnothing} S_j = \varnothing \qquad \bigcap_{j\in\varnothing} S_j = X.$$

We now return to the context of general maps $f : X \to Y$. Given a subset $A \subseteq Y$, we define its *preimage* under f as

$$f^*(A) = \{x \in X \mid fx \in A\}.$$

We would like to conceive of f^* as a map. Since it takes arguments and values in subsets it is a map of type $\mathcal{P}Y \to \mathcal{P}X$. We can restrict this map to singletons $\{y\}$ in Y. In this case, we abuse notation and write $f^* : Y \to \mathcal{P}X$. We use this map to define the *jectivity* properties of f; in particular, we say that f is

- *surjective* if for all $y \in Y$, $f^*(y)$ has *at least* 1 element
- *injective* if for all $y \in Y$, $f^*(y)$ has *at most* 1 element

These properties have an interpretation in the context of equations. An *equation*

$$f(x) = y$$

is merely a prompt, given a map f and a value $y \in Y$, to compute the preimage $f^*(y)$. We say that f has the

- *existence property* if for all $y \in Y$, there exists a solution to $f(x) = y$
- *uniqueness property* if for all $y \in Y$, any solution to $f(x) = y$ is unique

To write expressions for solutions to equations $[g \circ f](x) = z$ that involve map compositions $X \xrightarrow{f} Y \xrightarrow{g} Z$, we simply apply iterated preimages:

$$\mathcal{P}Z \xrightarrow{g^*} \mathcal{P}Y \xrightarrow{f^*} \mathcal{P}X$$

As an example, consider the equation $\tan^2(x) = 3$, which can be rewritten as

$$[\square^2 \circ \tan](x) = 3.$$

Before solving our equation, we compute the relevant reduced preimages

$$\tan^* : \mathbb{R} \to \mathcal{P}\mathbb{R} :: x \mapsto \{\arctan x + 2\pi n \mid n \in \mathbb{Z}\} =: \arctan x + 2\pi\mathbb{Z}$$
$$[\square^2]^* : \mathbb{R} \to \mathcal{P}\mathbb{R} :: x \mapsto \{\sqrt{x}, -\sqrt{x}\} =: \pm\sqrt{x}.$$

We now compute our solution set A:

$$\begin{aligned} A &= [(\tan)^* \circ (\Box^2)^*](3) \\ &= (\tan)^*(\pm\sqrt{3}) \\ &= \arctan(\pm\sqrt{3}) + 2\pi\mathbb{Z} \\ &= \pm\frac{\pi}{3} + 2\pi\mathbb{Z} \end{aligned}$$

Applying this to the context of a quadratic map q, its roots can be expressed as

$$[L_h^* \circ [\Box^2]^* \circ L_v^*](0).$$

We say that f is *bijective* when it is both surjective and injective. In this case, for each $y \in Y$, there is always precisely one element $x \in X$ such that $fx = y$. This allows us to construct a map $f^{-1} : Y \to X$ sending y to this unique element. In other terms, $f^*(y)$ is a singleton $\{x\}$ and, since it is precisely this sole element x to which y gets sent, we write $f^{-1}y = x$. We say the map f^{-1} is the *inverse* map to f. Given a map $f : X \to Y$, its inverse map $g : Y \to X$ is a map such that

$$\begin{aligned} g \circ f &= \mathbb{1}_X \\ f \circ g &= \mathbb{1}_Y. \end{aligned}$$

We say $f : X \to Y$ is *invertible*, or an *isomorphism* when it has an inverse map. We then write $f : X \xrightarrow{\cong} Y$ or, when we wish to speak directly about the sets in consideration, we write $X \cong Y$ and say that "X is isomorphic to Y." This language allows us to formalize our remark about singleton sets: $\star \cong \mathbf{1}$; or in general any pair of sets of the same cardinality, e.g. $\mathbb{B} \cong \mathbf{2}$.

Just as map composition generalizes numeric addition and multiplication, map inverse generalizes additive and multiplicative inverse.

$$\begin{aligned} (\Box + n) \circ (\Box + [-n]) &= \Box + 0 \\ (\Box \cdot n) \circ (\Box \cdot n^{-1}) &= \Box \cdot 1 \end{aligned}$$

In other words: $(\Box+n)^{-1} = (\Box+[-n])$ and $(\Box\cdot n)^{-1} = (\Box\cdot n^{-1})$. This also sheds light on the adage "you can't divide by 0"—the map $(\Box \cdot 0)$ amounts to the rule $x \mapsto 0$, which is surely neither injective nor surjective, and hence non-invertible.

This language inspires a reprisal of the fundamental theorems of calculus. We encode the derivative as $D : \mathcal{C}^\infty \to \mathcal{C}^\infty :: f \mapsto f'$, the integral as the map I in Equation (1), and the needed shifting adjustment as

$$\sigma : \mathcal{C}^\infty \to \mathcal{C}^\infty :: f \mapsto \lambda x.[f(x) - f(0)].$$

Then the first and second fundamental theorems can be rewritten as follows.

$$\begin{aligned} I \circ D &= \sigma \\ D \circ I &= \mathbb{1}_{\mathcal{C}^\infty} \end{aligned}$$

These equations encode precisely how the derivative and integral are "inverse" to each other: I is a right inverse but not a left inverse to D—it is however rather close to being a left inverse since it is only off by a shift. This shift is of course the consequence of the arbitrary constant that gets lost in the process of differentiating and hence, instead of being recovered, is replaced by a default constant $f(0)$.

The preimage notion also facilitates a more algebraic repackaging of set-builder notation. We can consider a predicate $\mathbf{P}$ on X as a map $\mathbf{P} : X \to \mathbb{B}$, in which case we should write $\mathbf{P}(x) = \top$ in place of just $\mathbf{P}(x)$ for "x satisfies $\mathbf{P}$." In turn, this added detail allows us to encode the set as a preimage of $\top$ under $\mathbf{P}$:

$$\mathbf{P}^*(\top) = \{x \in X \mid \mathbf{P}(x) = \top\}.$$

Aside from aesthetics, why would one care to do this? This notation throws in relief the correspondence between logical and set theoretic constructions. We list some examples in the following table, whose first two columns correspond to the informal and symbolic logical expressions.

P implies Q	$P \Rightarrow Q$	$P^*(\top) \subseteq Q^*(\top)$
P if and only if Q	$P \Leftrightarrow Q$	$P^*(\top) = Q^*(\top)$
P or Q	$P \vee Q$	$P^*(\top) \cup Q^*(\top)$
P and Q	$P \wedge Q$	$P^*(\top) \cap Q^*(\top)$

Making these relations explicit in the notation can allow for more succinct proofs via immediate symbolic reasoning. I'm of the belief that a goal of mathematical notation should be the excising of indirect, seemingly necessary, verbal fillers like "such that" from one's inner monologue.

We end the section by noting an important isomorphism.

Proposition 0.1. *The following is an isomorphism*

$$\lambda\varphi.\varphi^*(\top) : [X \to \mathbb{B}] \to \mathcal{P}X.$$

Proof. The inverse of this map is given by the map $\top_{(-)} : \mathcal{P}X \to [X \to \mathbb{B}]$, which maps a subset $S \subseteq X$ to the following so called *indicator* map,

$$\top_S : X \to \mathbb{B} :: x \mapsto \begin{cases} \top & x \in S \\ \bot & x \notin S \end{cases}$$

By construction $[\top_S]^*(\top) = S$ and $\top_{\varphi^*(\top)} = \varphi$. $\square$

1. Order Theory

We now define our first structured object.

Definition 1.1. A *preorder* $(P, \preceq)$ consists of the data

- a set P
- a map $\preceq: P \times P \to \mathbb{B}$, called a *relation*, writing $x \preceq y$ for $\preceq(x, y) = \top$,

satisfying the conditions

- $x \preceq x$ (reflexivity)
- $x \preceq y, y \preceq z \Rightarrow x \preceq z$ (transitivity)

We say that $(P, \preceq)$ is a *partially ordered set*, or *partial order*, or *poset* if furthermore

- $x \preceq y, y \preceq x \Rightarrow x = y$ (antisymmetry)

We say the poset $(P, \preceq)$ is a *total order* or *linear order* if in addition

- $x \preceq y$ or $y \preceq x$ (completeness)

We say that two preorders $(P, \preceq)$ and $(P', \preceq')$ are equal if and only if $P = P'$ and $\preceq = \preceq': P \times P \to \mathbb{B}$.

One family of total orders is given by numerical sets $\mathbf{n}, \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \ldots$ with the relation $\leq$. In a similar vein, since false formally implies true, i.e. $\bot \Rightarrow \top$, we have the total order $(\mathbb{B}, \Rightarrow)$. All finite cardinality total orders are essentially equivalent; or, as we shall soon define, isomorphic; to $(\mathbf{n}, \leq)$. Things become interesting in the case of infinite cardinality, as in the theory of *ordinals*, which we will not discuss.

It will be visually helpful—and soon formally meaningful—to represent a preorder $(P, \preceq)$ by a so called *Hasse Diagram.* This diagram is drawn by arranging the elements—often strategically—on the page and then drawing an arrow $x \to y$ to represent $x \preceq y$. For the sake of readability, when $x \preceq y$ and $y \preceq z$ we do not draw an arrow $x \to z$; rather, we take advantage of transitivity and draw the minimum number of arrows that imply the remaining ones, now visually encoded as a path of directed edges. Furthermore, by reflexivity, we never draw a loop from a vertex to itself. This can be absorbed into the path semantics via conceiving of there always being a length 0 path from a vertex to itself. This procedure of omitting implied arrows is called the *transitive reduction* of $(P, \preceq)$. For example, we can represent $(\mathbf{n}, \leq)$ as the diagram

$$0 \to 1 \to 2 \to \cdots \to n-1.$$

Using this language, we can draw any *directed graph*, i.e. a set of vertices (standing for elements) and a set of arrows or *directed edges* between them, and conceive of it as the Hasse Diagram for a preorder $(P, \preceq)$, where P is the set of vertices and $\preceq$ corresponds to the existence of a path of arrows. We call the procedure of taking a directed graph and conceiving of it as a poset its *transitive closure.* We can then define the transitive reduction, and hence Hasse Diagram, of $(P, \preceq)$ as the smallest directed graph whose transitive closure is $(P, \preceq)$. We can now use this practice to generate preorders via pictures. For example, the following diagram represents the simplest preorder that isn't a poset.

$$\begin{tikzcd} x \arrow[r, shift left] & y \arrow[l, shift left] \end{tikzcd}$$

For the time being, we will focus on *nonlinear* posets, or ones that aren't totally ordered. The simplest example of such is given by the following two posets.

$$x \to y \leftarrow z \qquad x \leftarrow y \to z$$

Note that if we reverse the arrows on one of these posets, we get the other poset. We then say the latter is the *opposite* preorder of the former, and vice versa. More formally, for a preorder $(P, \preceq)$, we write $(P, \preceq)^{\mathrm{op}}$ for its opposite preorder, given by precomposing $\preceq \colon P \times P \to \mathbb{B}$ with the map $\lambda(x, y).(y, x) : P \times P \to P \times P$. Or, equivalently: $x \preceq y$ in $(P, \preceq)$ if and only if $y \preceq x$ in $(P, \preceq)^{\mathrm{op}}$. Note that we have that $(-)^{\mathrm{op}}$ is an *involution*, i.e.

$$[(P, \preceq)^{\mathrm{op}}]^{\mathrm{op}} = (P \preceq).$$

We now introduce one of the most famous family of posets: $(\mathcal{P}X, \subseteq)$. We draw below the Hasse Diagram for the case of $X = \{a, b, c\}$.

(2)
$$\begin{tikzcd}
 & \{a,b,c\} & \\
\{a,b\} \arrow[ur] & \{a,c\} \arrow[u] & \{b,c\} \arrow[ul] \\
\{a\} \arrow[u] \arrow[ur] & \{b\} \arrow[ul] \arrow[ur] & \{c\} \arrow[ul] \arrow[u] \\
 & \varnothing \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Note that this forms a cube. This is for good reason: by $\mathcal{P}X \cong [X \to \mathbb{B}]$, which, if we compose with an isomorphism $\mathbb{B} \xrightarrow{\cong} \mathbf{2} :: \top \to 1$, a map $X \to \mathbf{2}$. Then, if we order the elements of X; i.e. choose an isomorphism $X \xrightarrow{\cong} \mathbf{n}$, for n the cardinality of X; we can view this map as assigning each subset of X a tuple of 0's and 1's. If we conceive of these tuples as representing coordinates in $\mathbb{R}^n$, then we precisely have the unit n-cube centered at the origin.

(3)
$$\begin{tikzcd}
 & (1,1,1) & \\
(1,1,0) \arrow[ur] & (1,0,1) \arrow[u] & (0,1,1) \arrow[ul] \\
(1,0,0) \arrow[u] \arrow[ur] & (0,1,0) \arrow[ul] \arrow[ur] & (0,0,1) \arrow[ul] \arrow[u] \\
 & (0,0,0) \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Given a preorder $(P, \preceq)$ and a subset $Q \subseteq P$, we can let Q inherit the preorder structure from P to form the *suborder* $(Q, \preceq)$. We now define two significant classes of suborders. Given an element $x \in P$, we define its *upper set* $\mathcal{U}_x$ and *lower set* $\mathcal{L}_x$ as follows.

$$\mathcal{U}_x(P, \preceq) = \{y \in P \mid x \preceq y\}$$
$$\mathcal{L}_x(P, \preceq) = \{y \in P \mid y \preceq x\}$$

When clear from context, we will suppress the $(P, \preceq)$ from the notation. An upper or lower set uniquely specifies an element. More formally, we have the following.

Proposition 1.2. *Let $x, y \in P$ for a poset $(P, \preceq)$. Let $\mathcal{U}_x$ and $\mathcal{U}_y$ be the respective upper sets for x and y, and suppose that $\mathcal{U}_x = \mathcal{U}_y$. Then we have that $x = y$.*

Proof. By reflexivity, $x \in \mathcal{U}_x, y \in \mathcal{U}_y$. Hence, supposing $\mathcal{U}_x = \mathcal{U}_y$, $x \in \mathcal{U}_y, y \in \mathcal{U}_x$. Then $x \preceq y$ and $y \preceq x$, and therefore, by antisymmetry, $x = y$. $\square$

Recalling the opposite construction, we remark that

$$\mathcal{L}_x(P, \preceq) = \mathcal{U}_x(P, \preceq)^{\mathrm{op}}.$$

We can conclude, via involutivity, that the same holds if we swap the place of $\mathcal{U}$ and $\mathcal{L}$. We will exploit this so called *duality* to derive facts about lower sets from facts about upper sets–for example, Proposition 1.2. We will hence focus our attention on upper sets. Let's redraw the Hasse diagram for $(P\{a, b, c\}, \subseteq)$, coloring red the directed edges belonging to $\mathcal{U}_{\{a\}}$ and blue the directed edges belonging to $\mathcal{U}_{\{b\}}$. In the situation where an edge belongs to both $\mathcal{U}_{\{a\}}$ and $\mathcal{U}_{\{c\}}$, we will—*exclusively* for the sake of readability—draw the edge twice, once in each color.

$$\begin{tikzcd}
 & \{a,b,c\} & \\
\{a,b\} \arrow[ur, red] & \{a,c\} \arrow[u, red, shift right] \arrow[u, blue, shift left] & \{b,c\} \arrow[ul, blue] \\
\{a\} \arrow[u, red] \arrow[ur, red] & \{b\} \arrow[ul] \arrow[ur] & \{c\} \arrow[ul, blue] \arrow[u, blue] \\
 & \varnothing \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd} \tag{4}$$

Consider the suborder $\mathcal{U}_{\{a\}} \cap \mathcal{U}_{\{c\}}$ with Hasse Diagram

$$\{a, c\} \to \{a, b, c\},$$

given by the suborder consisting of doubly colored edges in the above Hasse diagram. Observe that this suborder is precisely $\mathcal{U}_{\{a,c\}}$ and that, furthermore, $\{a, c\}$ is the union of the sets $\{a\}$ and $\{c\}$. Hence we have that $\mathcal{U}_{\{a\} \cup \{c\}} = \mathcal{U}_{\{a\}} \cap \mathcal{U}_{\{c\}}$ This holds true for any pair of subsets $S, T \subseteq X$ as elements in $(\mathcal{P}X, \subseteq)$; i.e

$$\mathcal{U}_{S \cup T} = \mathcal{U}_S \cap \mathcal{U}_T$$

We say that the set union $\cup$ is the *join*, denoted $\vee$, of $(\mathcal{P}, \subseteq)$.

Definition 1.3. Given a preorder $(P, \preceq)$ and two elements $x, y \in P$, their join $x \vee y$ is characterized by the following property. Given any element $t \in P$, we write

$$\frac{x \preceq t, y \preceq t}{\overline{x \vee y \preceq t}}$$

This diagram should be read as "given that which is above the line, we can deduce that which is below the line." The fact that the line is doubled means that the reverse also holds: "given that which is below the line, we can deduce that which is above the line." Hence this encodes the fact that a pair of relations $x \preceq t, y \preceq t$ is *equivalent data* to the single relation $x \vee y \preceq t$.

Since the statement $x \preceq t$ is equivalent to $t \in \mathcal{U}_x$, then $x \preceq t, y \preceq t$ is equivalent to $t \in \mathcal{U}_x \cap \mathcal{U}_y$ and $x \vee y \to t$ is equivalent to $t \in \mathcal{U}_{x\vee y}$. This implies that an equivalent characterization of the join can be given in terms of upper sets:

$$\mathcal{U}_{x\vee y} = \mathcal{U}_x \cap \mathcal{U}_y.$$

Given this definition of the join, we can define the dual concept, called the *meet* $x \wedge y$ of two elements $x, y \in P$, as the join of x and y in the opposite order $(P, \preceq)^{\mathrm{op}}$. From this, we can characterize the meet in terms of upper sets as

$$\mathcal{L}_{x\wedge y} = \mathcal{L}_x \cap \mathcal{L}_y,$$

or, alternatively, in a logical formulation, as

$$\frac{t \preceq x, t \preceq y}{\overline{t \preceq x \wedge y}}$$

This is our first example of a *universal property*. These are by some considered *the* object of study of category theory. They also always come in dual pairs.

Let's return to our example of the powerset order. Just as we characterized the set union order-theoretically as the join, we remark that the set intersection $\cap$ is the meet $\wedge$ in $(\mathcal{P}X, \subseteq)$. The line of thought is merely the dual of the above; but, given that a picture is worth a thousand words, we depict, by respectively coloring their lower sets red and blue, the fact that the meet of $\{a, c\}$ and $\{b, c\}$ is $\{c\}$, their intersection.

$$\begin{tikzcd}
 & & \{a,b,c\} & & \\
\{a,b\} \arrow[urr] & & \{a,c\} \arrow[u] & & \{b,c\} \arrow[ull] \\
\{a\} \arrow[u] \arrow[urr, blue] & & \{b\} \arrow[ull] \arrow[urr, red] & & \{c\} \arrow[u, red] \arrow[ull, blue] \\
 & & \varnothing \arrow[ull, blue] \arrow[u, red] \arrow[urr, blue] \arrow[urr, red, shift right] & &
\end{tikzcd} \tag{5}$$

We remark that in the context of conceiving of these diagrams as cubes, that the upper and lower sets are faces and their intersection is an edge. We will return to this idea in our forthcoming reprisal of linear algebra.

Note that, if $x \preceq y$ in a poset $(P, \preceq)$ then $\mathcal{U}_y \subseteq \mathcal{U}_x$ and $\mathcal{L}_x \subseteq \mathcal{L}_y$. This implies that $x \vee y = y$ and $x \wedge y = x$. Then, in the context of a total order, since $x \preceq y$ or $y \preceq x$, the join and meet concepts simply reduce to determining which of the arguments is respectively bigger and smaller. For example, in the context of $(\mathbb{N}, \leq)$, we have that join and meet reduce to maximum and minimum.

$$n \vee m = \max(n, m)$$
$$n \wedge m = \min(n, m)$$

Let's now consider another family of examples. Let $n \in \mathbb{N}$ and denote by $\langle n \rangle$ the set of $k \in \mathbb{N}$ for which $k|n$, i.e. which divide n. Then we have a poset $(\langle n \rangle, |)$.

We now draw the Hasse diagram in the case of $n = 30$, coloring red and blue the upper set of 2 and 5, respectively.

$$\begin{tikzcd}
& 30 & \\
6 \arrow[ur, red] & 10 \arrow[u, red, shift right] \arrow[u, blue, shift left] & 15 \arrow[ul, blue] \\
2 \arrow[u, red] \arrow[ur, red] & 3 \arrow[ul] \arrow[ur] & 5 \arrow[ul, blue] \arrow[u, blue] \\
& 1 \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd} \tag{6}$$

Note that, up to relabelling vertices, this is the same Hasse diagram with that of $(\mathcal{P}\{a, b, c\}, \subseteq)$. The presence of the factors $2, 3, 5$ plays the same role as the inclusion of the elements a, b, c. By the same reasoning as before, we note the join $2 \vee 5 = 10$. But we also know from above that $10 \vee 5 = 10$. This is none other than the least common multiple or lcm. Dually, the meet in this poset is the greatest common divisor, or gcd.

Note that, when $p^2 | n$, such as with $n = 12$, $(\langle n \rangle, |))$ is no longer a cube:

$$\begin{tikzcd}
& 12 & \\
4 \arrow[ur] & & 6 \arrow[ul] \\
2 \arrow[u] \arrow[urr] & & 3 \arrow[u] \\
& 1 \arrow[ul] \arrow[ur] &
\end{tikzcd} \tag{7}$$

Not every poset enjoys meets and joins for all of its pairs. For example, the poset on $\{w, x, y, z\}$ with the following Hasse diagram lacks many meets and joins.

$$w \to x \qquad y \to z$$

Definition 1.4. The poset $(P, \preceq)$ is a *lattice* if for all pairs $x, y \in P$ there exists both join $x \vee y$ and meet $x \wedge y$.

Thus far we have considered joins and meets as binary operators. These can be generalized to operators that accept subsets $S \subseteq P$ as arguments.

Definition 1.5. Given a poset $(P, \preceq)$ and a subset $S \subseteq P$, we define its *least upper bound* or *supremum*, denoted $\sup S$, as the unique elements in P satisfying the following condition.

$$\begin{array}{c} \sup S \preceq z \\ \hline\hline \forall s \in S, s \preceq z \end{array}$$

Dually, we define its *greatest lower bound* or *infimum*, denoted $\inf S$, as the unique elements in P satisfying the following condition.

$$\begin{array}{c} z \preceq \inf S \\ \hline\hline \forall s \in S, z \preceq s \end{array}$$

In the special case that $S = \{x, y\}$, we have that

$$\sup S = x \vee y$$
$$\inf S = x \wedge y$$

In the language of upper and lower sets, we can characterize sup and inf as follows.

$$\mathcal{U}_{\sup S} = \bigcap_{s \in S} \mathcal{U}_s$$
$$\mathcal{L}_{\inf S} = \bigcap_{s \in S} \mathcal{L}_s$$

In the finite case, we will see—in the first assignment—that the existence of joins and meets implies the existence of suprema and infima. When S is infinite—or when $S = \varnothing$ and P is infinite—we will see that this is no longer the case. We say that a lattice closed under both sup and inf is *complete*. We say $(Q, \preceq)$ is a *completion* of $(P, \preceq)$ when P is a sub-order of Q and Q is complete. Famously, $(\mathbb{R}, \leq)$ was originally defined as the smallest completion of $(\mathbb{Q}, \leq)$.

Just as maps are the "verb" to the sets' "noun," orders too have a map-notion.

Definition 1.6. A *monotone map* $(P, \preceq) \to (Q, \sqsubseteq)$ consists of

- a map $f : P \to Q$

such that

- $x \preceq y \Rightarrow fx \sqsubseteq fy$.

An *antitone map* : $(P, \preceq) \to (Q, \sqsubseteq)$ is a monotone map : $(P, \preceq)^{\mathrm{op}} \to (Q, \sqsubseteq)$.

We now list some examples.

(1) $(\square + n), (\square \cdot t), (\square^k) : (\mathbb{N}, \leq) \to (\mathbb{N}, \leq)$ for $n, k, t \in \mathbb{N}$
(2) $(\square \cdot t), (\square^k) : (\mathbb{N}, |) \to (\mathbb{N}, |)$ for $t, k \in \mathbb{N}$ since $a|b \Rightarrow (ta)|(tb), a^k|b^k$.
(3) $\lambda n.n : (\mathbb{N}, |) \to (\mathbb{N}, \leq)$ since $a|b \Rightarrow a \leq b$.
(4) $(-) \cup T, (-) \cap T : (\mathcal{P}X, \subseteq) \to (\mathcal{P}X, \subseteq)$ for $T \subseteq X$.
(5) the *set complement*, given by the X elements *not* in S:

$$(-)^c : (\mathcal{P}X, \subseteq)^{\mathrm{op}} \to (\mathcal{P}X, \subseteq) :: S \mapsto S^c = \{x \in X \mid x \notin S\},$$

since $A \subseteq B \Rightarrow B^c \subseteq A^c$.
(6) given a map $f : X \to Y$, its preimage $f^* : (\mathcal{P}Y, \subseteq) \to (\mathcal{P}X, \subseteq)$, since

$$A \subseteq B \Rightarrow f^*A \subseteq f^*B.$$

We now list some non-examples.

(1) $\square^2 : (\mathbb{Z}, \leq) \to (\mathbb{Z}, \leq)$ since $(-3) \leq (-1)$ but $(-1)^2 = 1 \leq 9 = (-3)^2$.
(2) $(\square + k) : (\mathbb{N}, |) \to (\mathbb{N}, |)$ for $k \neq 0$, since $1|p - k$ but not $1 + k|p$ for prime p.
(3) $\lambda n.n : (\mathbb{N}, \leq) \to (\mathbb{N}, |)$ since $p \leq q$ but not $p|q$ for primes p, q.

We say two posets $(P, \preceq), (Q, \sqsubseteq)$ are *isomorphic* if there exist monotone maps $f : (P, \preceq) \to (Q, \sqsubseteq)$ and $g : (Q, \sqsubseteq) \to (P, \preceq)$ such that $g \circ f = \mathbb{1}_P$ and $f \circ g = \mathbb{1}_Q$. Although a monotone map often has no inverse, it may sometimes possess another kind of meaningful map going in the reverse direction.

Definition 1.7. We say the pair (L, R) of monotone maps $L : (P, \preceq) \to (Q, \sqsubseteq)$ and $R : (Q, \sqsubseteq) \to (P, \preceq)$ form a *Galois connection* when:

$$\frac{Lx \sqsubseteq y}{\overline{x \preceq Ry}}$$

We call L the *left Galois connection* of R and R the *right Galois connection* of L.

It is worth broadcasting a caution that this definition is emphatically asymmetric in L and R. Let's consider an example. Define

$$i : (\mathbb{Z}, \leq) \to (\mathbb{Q}, \leq) :: n \mapsto n$$

as the map merely *retypes* an integer as a rational number, without changing what we conceive of as its value. How do we go back the other way; i.e. get an integer from a rational? There's two particularly available options:

$$\text{floor} : \mathbb{Q} \to \mathbb{Z} :: q \mapsto \lfloor q \rfloor$$
$$\text{ceiling} : \mathbb{Q} \to \mathbb{Z} :: q \mapsto \lceil q \rceil,$$

Which are defined, respectively, as the largest integer less than q and the smallest integer greater than q. Notice that these definitions take on a similar linguistic character as those for meets and joins. This is no coincidence, as we shall shortly investigate further. These statements can actually be reformulated in terms of the following Galois connections.

$$\frac{i(n) \leq q}{\overline{n \leq \lfloor q \rfloor}}$$

$$\frac{q \leq i(n)}{\overline{\lceil q \rceil \leq n}}$$

In other words, (i, floor) and $(\text{ceiling}, i)$ are Galois connections. The following—our first!—theorem justifies why we care about Galois connections.

Theorem 1.8. *Let $(P, \preceq)$ and $(Q, \sqsubseteq)$ be preorders and $(L : P \to Q, R : Q \to P)$ be a Galois connection. Then L preserves suprema and R preserves infima.*

Proof.

$$\begin{aligned} z \in \mathcal{U}_{L(\sup S)} &\Leftrightarrow L(\sup S) \sqsubseteq z \\ &\Leftrightarrow \sup S \preceq Rz \\ &\Leftrightarrow \forall s \in S, s \preceq Rz \\ &\Leftrightarrow \forall s \in S, Ls \sqsubseteq z \\ &\Leftrightarrow \forall s' \in L(S), s' \sqsubseteq z \\ &\Leftrightarrow \sup L(S) \sqsubseteq z \\ &\Leftrightarrow z \in \mathcal{U}_{\sup L(S)} \end{aligned}$$

Therefore $\mathcal{U}_{L(\sup S)} = \mathcal{U}_{\sup L(S)}$, and hence, by Proposition 1.2, $L(\sup S) = \sup L(S)$. Applying $(-)^{\text{op}}$ gives the dual result $R(\inf S) = \inf R(S)$ □

Applying this to our above example, we have that

$$\lceil \sup S \rceil = \sup \lceil S \rceil$$
$$\lfloor \inf S \rfloor = \inf \lfloor S \rfloor$$

In the binary case, where the supremum and infimum respectively reduce to the maximum and mimum of a pair of numbers, it turns out that both ceil and floor preserve both maxima and minima. These two other facts—that ceil preserves minima and floor preserves maxima—turn out to be incidental in a way distinct from those implied by our Galois connections. To see this, we note that this breaks down in the infinite case. Let $S = \{\frac{1}{n}\}_{n\in\mathbb{N}}$, then we have that $\inf S = 0$, and hence

$$\begin{aligned}\lceil \inf\{\tfrac{1}{n}\}_{n\in\mathbb{N}}\rceil &= \lceil 0\rceil \\ &= 0,\end{aligned}$$

but, in contrast

$$\begin{aligned}\inf \lceil\{\tfrac{1}{n}\}_{n\in\mathbb{N}}\rceil &= \inf 1 \\ &= 1.\end{aligned}$$

Note that the Galois Connections also imply that i preserves both ceilings and floors. On one hand, since i does not change its arguments' "values," this may be thought to have been obvious. This turns out, however, to be substantive. Consider, in contrast, the map $\lambda n.n : (\mathbb{N}, |) \to (\mathbb{N}, \leq)$ also leaves its arguments' values unchanged—the immersion, however, into a new ordering drastically alters suprema and infima.

We now consider an example of a Galois connection involving antitone maps, or an antitone Galois connection. Given a set X, we have the antitone complement map $\mathcal{P}X \to \mathcal{P}X$. This map can be seen as a monotone map of type either $\mathcal{P}X^{\mathrm{op}} \to \mathcal{P}X$ or $\mathcal{P}X \to \mathcal{P}X^{\mathrm{op}}$. For the sake of being explicit, we respectively name these two maps comp and comp′. It turns out that (comp, comp′) form a Galois connection. This says that the following equivalence holds.

$$\frac{\mathrm{comp}(S) \subseteq T}{\overline{S \subseteq \mathrm{comp}'(T)}}$$

To see this, replace these type-sensitive names with the simpler $\overline{\square}$ notation:

$$\frac{\overline{S} \subseteq T}{\overline{S \subseteq \overline{T}}}$$

This can be seen by applying the complement to the top and noting that $\overline{\overline{S}} = S$. Applying Theorem 1.8, and using the $\overline{\square}$ notation, we see that

$$\overline{S \cup T} = \overline{S} \cap \overline{T}$$
$$\overline{S \cap T} = \overline{S} \cup \overline{T}.$$

Note that, since the complement is antitone, it swaps meets and joins in input and output. The reader may already be familiar with these as the *De Morgan Laws.* Given a Galois connection (L, R) between lattices, we will henceforth refer to the following resultant identities as the *De Morgan Laws* for (L, R).

$$L(x \vee y) = Lx \vee Ly$$
$$R(x \wedge y) = Rx \wedge Ry.$$

We end the section by exploring another consequence of Galois connections.

Proposition 1.9. *If (L, R) is a Galois connection on posets, then*

$$LRL = L \qquad RLR = R$$

Proof.

$$\begin{aligned}\text{reflexivity } &\Rightarrow RLx \preceq RLx \\ &\Rightarrow LRLx \preceq Lx\end{aligned}$$

$$\begin{aligned}\text{reflexivity } &\Rightarrow Lx \preceq Lx \\ &\Rightarrow x \preceq RLx \\ &\Rightarrow Lx \preceq LRLx\end{aligned}$$

Using antisymmetry yields the first identity. Duality yields the second. □

A corollary of this is that LR and RL are both idempotent, i.e.

$$LRLR = LR \qquad RLRL = RL.$$

In the context of floors and ceilings, this justifies the well known facts

$$\lceil\lceil x\rceil\rceil = \lceil x\rceil \qquad \lfloor\lfloor x\rfloor\rfloor = \lfloor x\rfloor.$$

By rearranging reflexivity, we also have that

$$x \preceq RLx \qquad LRx \preceq x.$$

We call idempotent monotone operators like RL *closure* operators, and those like LR *interior* operators. Usually the former is the preferred notion. Closure operators abound in mathematics and formalize a process that chooses for some element in a poset the least larger element that satisfies some property. For example, if V is some k-vector space, and we look at $\mathcal{P}V$, we may wish to ask: given $S \subseteq V$, what is the smallest subspace U containing S? This is precisely the k-linear span:

$$kS = \{\textstyle\sum_{i=1}^{n} \alpha_i s_i \mid \alpha_i \in k, s_i \in S\}.$$

In fact, this closure operator arises from a Galois connection! Let $\mathcal{S}V$ be the suborder of $\mathcal{P}V$ consisting of subspaces of V and let $i : \mathcal{S}V \to \mathcal{P}V$ be the inclusion and $k : \mathcal{P}V \to \mathcal{S}V$ be the k-linear span, thought of as having codomain subspaces. Then (k, i) is a Galois connection, i.e.

$$\frac{kS \subseteq U}{\overline{S \subseteq i(U)}}$$

This fact—that S is a subset of some subspace U just if its span is a subset of U—is, given our theory, precisely why the linear span is a closure operator.

2. Categories, Functors, and Natural Transformations

We'll jump straight into the definition.

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

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

satisfying the conditions

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

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

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

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

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

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

(2) any preorder $(P, \preceq)$ can be seen as a category $\overline{P}$ whose objects are again elements of P, but whose arrows $x \to y$ correspond precisely to statements $x \preceq y$, as was visualized in our Hasse diagrams. More formally, $\overline{P}$ can be characterized as follows.
   - $|\overline{P}| = P$
   - $\overline{P}(x, y) = \begin{cases} \star & x \preceq y \\ \varnothing & x \npreceq y \end{cases}$

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

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

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

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

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

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

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

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

One can conceive of $\mathbf{Pos}$ as a subcategory of $\mathbf{Pre}$, $\mathbf{Tot}$ as a subcategory of $\mathbf{Pos}$, $\mathbf{Com}$ as a subcategory of $\mathbf{Mon}$, and, as we shall discuss further, $\mathbf{Set}$ as a subcategory of $\mathbf{Rel}$.

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

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

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

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

satisfying the following conditions

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

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

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

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

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

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

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

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

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

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

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

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

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

Said otherwise, since a relation $X \times Y \to \mathbb{B}$ is, by Proposition 0.1, equivalent to a subset of $X \times Y$ corresponding to the points sent to true. In this case, this is precisely the set of points $\{(x, fx)\} \subseteq X \times Y$, which is the familiar notion of a graph of a function, most famously in the case of drawing curves in the 2 dimensional plane $\mathbb{R}^2$ as a way to represent maps $\mathbb{R} \to \mathbb{R}$.

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

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

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

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

A rather similar procedure can be considered for monoids.

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

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

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

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

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

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

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

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

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

via the behavior

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

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

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

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

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

via the behavior

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

essentially wide if for all $Z \in |\mathcal{D}|$, there is some $X \in |\mathcal{C}|$ for which $FX \cong Z$.

This condition is also called *essentially surjective on objects.* We drop the "essentially" from both names if the isomorphism is replaced by an equality. The up-to-isomorphism notion however will as usual play a more fundamental role.

We say the functor F is:

full if $\forall X, Y : F_{X,Y} : \mathcal{C}(X,Y) \to \mathcal{D}(X,Y)$ is surjective

faithful if $\forall X, Y : F_{X,Y} : \mathcal{C}(X,Y) \to \mathcal{D}(X,Y)$ is injective

When F is both full and faithful, we call it *fully faithful.* These functors satisfy the following useful property.

Proposition 2.10. *If $F : \mathcal{C} \to \mathcal{D}$ is fully faithful, then $FX \cong FY$ implies $X \cong Y$.*

Proof. By assumption, there are arrows

- $\alpha : FX \to FY$
- $\beta : FY \to FX$

satisfying

- $\beta \circ \alpha = \mathbb{1}_{FX}$
- $\alpha \circ \beta = \mathbb{1}_{FY}$

Since $F_{X,Y}$ and $F_{Y,X}$ are bijections, we have unique preimages to α, β, $\mathbb{1}_{FX}$, $\mathbb{1}_{FY}$:

- $\alpha' : X \to Y$
- $\beta' : Y \to X$
- $\mathbb{1}_X : X \to X$
- $\mathbb{1}_Y : Y \to Y$

We wish to prove that the following compositions are identities.

- $\beta' \circ \alpha'$
- $\alpha' \circ \beta'$

By functoriality, the image of these maps under F are precisely

- $\beta \circ \alpha = \mathbb{1}_{FX}$
- $\alpha \circ \beta = \mathbb{1}_{FY}$

These are also the images of $\mathbb{1}_X$ and $\mathbb{1}_Y$, which, by injectivity, force the equalities:

- $\beta' \circ \alpha' = \mathbb{1}_X$
- $\alpha' \circ \beta' = \mathbb{1}_Y$

Thus establishing the isomorphism $X \cong Y$. □

This can be conceived of as a kind of essential injectivity on objects, although this intuition is insufficiently well behaved to be deployed formally.

There is an interesting, albeit non domain-general, interpretation of these properties in terms of forgetful functors. Given a functor $F : \mathcal{C} \to \mathcal{D}$, interpreted as a forgetful functor, we say that F forgets

property if it fails to be essentially wide

structure if it fails to be full

stuff if it fails to be faithful

3. Universality and Representability

Let $\mathcal{C}$ be a category. We now identify some special, or *universal*, objects of $\mathcal{C}$.

Definition 3.1. We say 0 is the *initial object* of $\mathcal{C}$ if for any $\mathcal{C}$-object—including itself—X, there exists a unique arrow $0 \to X$. Dually, we say 1 is the *terminal object* of $\mathcal{C}$ if for any $\mathcal{C}$-object X, there exists a unique arrow $X \to 1$.

By duality, an initial object in $\mathcal{C}$ is a terminal object in $\mathcal{C}^{\mathrm{op}}$ and vice versa. When $\mathcal{C}$ is thin, i.e. a preorder, the bottom $\perp$ is the initial object and top $\top$ is the terminal object. From this observation, we remark that not all categories have initial or terminal objects—for example, the poset $(\mathbb{Z}, \leq)$, thought of as a thin category, has neither top nor bottom and hence neither initial nor terminal object. We often denote the unique arrow guaranteed by universality via a dashed line:

$$\begin{tikzcd} 0 \arrow[r, dashed] & S \end{tikzcd}$$

The use of the symbols 0 and 1 for generic initial and terminal objects is inspired by the fact that $\mathbf{0} = \varnothing$ and $\mathbf{1} \cong \star$ are respectively the initial and terminal objects in **Set**. The latter fact is easier to see: given a set S, a map $S \to \star$ consists of a choice of element in $\star$ for every $s \in S$—but there can be only one such choice: the sole element $* \in \star$. In turn, one can conceptualize the fact that $\varnothing$ is the initial object in **Set** by considering the fact that a map $\varnothing \to S$ consists of a choice for every element in $\varnothing$—but there are no such elements, and hence no choice needs to be made. Said otherwise, each element of the domain poses a challenge: to choose a single codomain element associated with it. In the case that there are no such challenges, we say—via some combination of philosophy and convention—that there is hence precisely one map that is gifted to us by a lack of choice in the matter. This is in some sense analogous to the situation in which we define nullary operators to be units.

In the context of **Pre**, the initial and terminal objects are, just as in **Set**, $\varnothing$ and $\star$, equipped with the unique ordering structure on them. This turns out to be no coincidence: **Pre**—and similarly **Pos**—inherit much of their "universal structure" from **Set**. This will be elaborated further in the subsequent section.

In both **Mon** and $\mathbf{Vect}_k$ we have a so called *zero object*—i.e. an object that is both terminal and initial. These are respectively given by the one element monoid $\{e\}$ and the zero vector space $\{0\}$. At the risk of mild notational overload, we will denote both of these by 0. These are terminal objects for the same reason that $\star$ is a terminal object in $\mathcal{C}$—every homormorphism to the one element structure maps the entire codomain to that element. Just as above, this arises from the fact that half of the universal structure of **Set** is inherited by **Mon** and $\mathbf{Vect}_k$. In turn, however, the initial object is not the empty set! This is because both monoids and vector spaces require unit elements. The arrows from these one element structures to any other structure are unique by virtue of the fact that these arrows by definition must preserve unit elements, and hence uniquely send the sole domain element to the unit of the codomain.

We now present the first special property of universal objects.

Proposition 3.2. *Let* 0 *and* $0'$ *be initial objects of* $\mathcal{C}$*. Then* $0 \cong 0'$.

Proof. By initiality, there are unique arrows $0 \to 0$ and $0' \to 0'$, which must be the identity arrows $\mathbb{1}_0$ and $\mathbb{1}_{0'}$. We also have unique arrows $\varphi : 0 \to 0'$ and $\varphi' : 0' \to 0$.

This yields arrows $\varphi \mathbin{/\!\!/} \varphi' : 0 \to 0$ and $\varphi' \mathbin{/\!\!/} \varphi : 0' \to 0$, which, by uniqeueness, must be the respective identity arrows. $\square$

By duality, the same fact holds for terminal objects. In fact, we will see throughout the section that any universal construction can be cast as an initial object in some category, and hence enjoys the result of this theorem.

As the title suggests, we will also consider a second approach, called representability, towards the study of categories. In particular, we say a functor $F : \mathcal{C} \to \mathbf{Set}$ is *representable* if there is a natural isomorphism $F \cong \mathcal{C}(c, -)$ for some $\mathcal{C}$-object c. Dually, we say a functor $F : \mathcal{C}^{\text{op}} \to \mathbf{Set}$ is representable if there is a natural isomorphism $F \cong \mathcal{C}(-, c)$ for some $\mathcal{C}$-object c. In both of these cases, we say that c *represents* the relevant functor. In turn, the functor that an object represents will often provide a semantic interpretation for the object. A particularly important example of reprsentable functor is the identity functor $\mathbb{1}_{\mathbf{Set}} : \mathbf{Set} \to \mathbf{Set}$. Recalling that an element $s \in S$ is equivalent to the arrow $\lambda * .s : \star \to S$, we have that $S \cong \mathbf{Set}(\star, S)$, and hence that $\mathbb{1}_{\mathbf{Set}} \cong \mathbf{Set}(\star, -)$.

Universal constructions always have an interpretation in the language of representability. In the case of initial and terminal objects, this may seem a little dull. In particular, consider the constant functor $\mathcal{C} \to \mathbf{Set}$ that sends every object to $\star$ and every arrow to $\mathbb{1}_\star$. If $\mathcal{C}$ has an initial object 0, then this functor is naturally isomorphic to $\mathcal{C}(0, -)$ since by definition the set of arrows $0 \to X$ for any X is a singleton. Dually, the constant functor $\mathcal{C}^{\text{op}} \to \mathbf{Set}$ with the same behavior is naturally isomorphic to $\mathcal{C}(-, 1)$. The contravariance, although not necessary to the well definedness of this particular functor, is stiuplated to respect the fact that $\mathcal{C}(-, c)$ is generically contravariant.

Representable functors are special because of the *Yoneda Lemma*, the most famous result in category theory.

Theorem 3.3. *Let $\mathcal{C}$ be a category, $c \in |\mathcal{C}|$, and $F : \mathcal{C} \to \mathbf{Set}$. Then there is a natural bijection*

$$\mathbf{Cat}(\mathcal{C}(c, -), F) \cong Fc$$

given by mapping a natural transformation $T : \mathcal{C}(c, -) \to F$ to $T_c(\mathbb{1}_c) \in Fc$.

Proof. Let $\Phi : \mathbf{Cat}(\mathcal{C}(c, -), F) \to Fc :: T \mapsto T(\mathbb{1}_c)$. We will now construct a map $\Psi : F(c) \to \mathbf{Cat}(\mathcal{C}(c, -), F)$ that satisfies $\Phi \circ \Psi = \mathbb{1}_{Fc}$ and $\Psi \circ \Phi = \mathbb{1}_{\mathbf{Cat}(\mathcal{C}(c,-),F)}$.

We define Ψ to be a valid inverse by mapping an element x of Fc to a natural transformation T for which $T_c(\mathbb{1}_c) = x$. We now show that this uniquely defines a natural transformation. Since T consists of component maps $T_{c'} : \mathcal{C}(c, c') \to Fc$ for all c', it suffices to define $T_{c'}(f)$ for all $f \in \mathcal{C}(c, c')$. Then, letting $f \in \mathcal{C}(c, c')$, the naturality condition states that the following square must commute.

$$\begin{tikzcd}
\mathcal{C}(c,c) \arrow[r, "T_c"] \arrow[d, "- /\!\!/ f"'] & Fc \arrow[d, "Ff"] \\
\mathcal{C}(c,c') \arrow[r, "T_{c'}"'] & Fc'
\end{tikzcd}$$

Tracing the two paths of the identity arrow $\mathbb{1}_c \in \mathcal{C}(c, c)$, we have:

$$\to\downarrow : \mathbb{1}_c \mapsto T_c(\mathbb{1}_c) = x \mapsto (Ff)x$$
$$\downarrow\to : \mathbb{1}_c \mapsto f \mapsto T_{c'}(f)$$

This then forces us to define $T_{c'}(f) = (Ff)x$. $\square$

The reader is encouraged to check that this bijection is natural in c. Furthermore, this bijection is natural in $\mathcal{C}$ and F, but this involves slightly more mental and notational contortion to articulate. Note that, were we to apply the Yoneda Lemma to $\mathcal{C}^{\mathrm{op}}$, we would have that for any $F : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ and $c \in |\mathcal{C}|$:

$$\mathbf{Cat}(\mathcal{C}^{\mathrm{op}}(c,-),F) = \mathbf{Cat}(\mathcal{C}(-,c),F) \cong Fc.$$

The Yoneda Lemma can be used to answer questions like "what are the natural maps $X \to \mathbf{Set}(S,X)$?" Said both more suggestively and formally: what are the natural transformations $\mathbb{1}_{\mathbf{Set}} \cong \mathbf{Set}(\star,-) \to \mathbf{Set}(S,-)$? The Yoneda Lemma states that these would be in natural bijection with elements of the set $\mathbf{Set}(S,\star)$, which, by the terminality of $\star$, form a singleton given by the constant map. Hence the only natural transformation $T : \mathbb{1}_{\mathbf{Set}} \to \mathbf{Set}(S,-)$ would be defined by extending $T_\star : \star \to \mathbf{Set}(S,\star) :: \mathbb{1}_\star \mapsto \lambda s.* : S$. Hence, applying the bijection given by the Yoneda Lemma, we have, for a set X and X-element x, i.e. arrow $x : \star \to X$, that:

$$T_X(x) = \mathbf{Set}(S,x) : \mathbf{Set}(S,\star) \to \mathbf{Set}(S,X) :: f \mapsto x.$$

Said more plainly, a natural map $\mathbf{Set}(\star,X) \to \mathbf{Set}(S,X)$ is just a precomposition with the map $x : \star \to S$, i.e. a map that sends the element $x \in X$ to the constant map $\lambda s.x : S \to X$. This then allows us to conclude that the only natural map—i.e. the only map that can be defined generically, with no reference to the sets involved—$X \to \mathbf{Set}(S,X)$ is the map that takes elements $x \in X$ to constant maps $S \to X$ that send each S-element to x.

This example involved the case that the functor F was itself representable. In this context, the Yoneda Lemma has another meaningful interpretation. Define the functor $\mathcal{Y}$, called the *Yoneda embedding*, of type $\mathcal{C}^{\mathrm{op}} \to \mathbf{Cat}(\mathcal{C},\mathbf{Set})$ as follows. For an object c, define $\mathcal{Y}c = \mathcal{C}(c,-)$. For an arrow $\varphi : c \to c'$, define the natural transformation $\mathcal{Y}\varphi : \mathcal{C}(c',-) \to \mathcal{C}(c,-)$ via the components

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

Then, the Yoneda lemma states that

$$\mathbf{Cat}(\mathcal{Y}c',\mathcal{Y}c) = \mathbf{Cat}(\mathcal{C}(c',-),\mathcal{C}(c,-)) \cong \mathcal{C}(c,c').$$

This can be interpreted as the statement that the Yoneda embedding $\mathcal{Y}$ is fully faithful; i.e. that $\mathcal{Y}_{c',c} : \mathcal{C}^{\mathrm{op}}(c',c) = \mathcal{C}(c,c') \to \mathbf{Cat}(\mathcal{Y}c',\mathcal{Y}c)$ is always a bijection. Recall that this means that $\mathcal{Y}$ has the property that if $\mathcal{Y}c \cong \mathcal{Y}c'$, then $c \cong c'$. This has the deep implication that an object is determined up to isomorphism by the functor it represents. This truth is in some sense why category theory contributes insight to mathematics beyond mere bookkeeping: rather, it allows one to speak of mathematical structures purely in terms of their relationships to all other structures of their kind, as opposed to relying on what they're "made of." As an example of this principle, consider the case of initial objects: since these always represent the constant set-valued functor of value $\star$, they must all be isomorphic.

Let's now consider a more interesting dual pair of universal constructions. We motivate these with a case study in the context of $\mathbf{Set}$. Consider a map $\varphi : S \to X \times Y$; for example, perhaps S consists of the set of points in some island, X and Y respectively denote the sets of possible values for temperature and pressure, and φ assigns to each point the ordered pair of air temperature and pressure at the altitude of an average human height. We can easily extract from this map two maps φ_X and φ_Y, which respectively report the temperature and the pressure. Intuitively, φ_X merely forgets the pressure data and φ_Y merely forgets the temperature data.

More formally, we can define $\varphi_X = \varphi \mathbin{/\!\!/} \pi_X$, where we call π_X the *projection* map, and define it by the rule $(x, y) \mapsto x$. Doing the same for Y, we note the following:

$$\varphi s = (\varphi_X(s), \varphi_Y(s))$$

Said otherwise, from a map $\varphi \in \mathbf{Set}(S, X \times Y)$, we can extract the pair of maps $(\varphi \mathbin{/\!\!/} \pi_X, \varphi \mathbin{/\!\!/} \pi_Y) \in \mathbf{Set}(S, X) \times \mathbf{Set}(S, Y)$, and, from a pair of maps $(\varphi_X, \varphi_Y) \in \mathbf{Set}(S, X) \times \mathbf{Set}(S, Y)$, we can extract the map $\lambda s.(\varphi_X s, \varphi_Y s) \in \mathbf{Set}(S, X \times Y)$. Furthermore, this correspondence yields a bijection

$$\mathbf{Set}(S, X \times Y) \cong \mathbf{Set}(S, X) \times \mathbf{Set}(S, Y).$$

Diagrammatically, this is often expressed as follows:

$$\begin{tikzcd}
 & S \arrow[d, dashed, "\varphi"] \arrow[ddl, "\varphi_X"'] \arrow[ddr, "\varphi_Y"] & \\
 & X \times Y \arrow[dl, "\pi_X"] \arrow[dr, "\pi_Y"'] & \\
X & & Y
\end{tikzcd}$$

The dashed arrow is read as follows: given a pair $(\varphi_X : S \to X, \varphi_Y : S \to Y)$, there exists a unique arrow $\varphi : S \to X \times Y$ making the diagram commute. This diagram also states that—following the existence of compositions—that, given a φ, composing it with the projections yields a pair $(\varphi_X : S \to X, \varphi_Y : S \to Y)$. The bijection and the diagram respectively characterize the universality and representability features of the Cartesian Product.

More generally, given a category $\mathcal{C}$ and two objects X_i for $i = 0, 1$, the *product* $X_0 \times X_1$, if it exists, is an object equipped with projection arrows $\pi_i : X_0 \times X_1 \to X_i$ for $i = 0, 1$, such that, given another object S and a pair of arrows $\varphi_i : S \to X_i$ for $i = 0, 1$, there exists a unique arrow $\varphi : S \to X_0 \times X_1$ making the following diagram in $\mathcal{C}$ commute:

$$\begin{tikzcd}
 & S \arrow[d, dashed, "\varphi"] \arrow[ddl, "\varphi_0"'] \arrow[ddr, "\varphi_1"] & \\
 & X_0 \times X_1 \arrow[dl, "\pi_0"] \arrow[dr, "\pi_1"'] & \\
X_0 & & X_1
\end{tikzcd}$$

The product $X_0 \times X_1$ then satisfies the natural bijection:

$$\mathcal{C}(-, X_0 \times X_1) \cong \mathcal{C}(-, X_0) \times \mathcal{C}(-, X_1).$$

We can encode this even more tightly as a terminal object or a representable functor. Letting X_0, X_1 be objects of $\mathcal{C}$, define the category $\mathcal{C}/(X_0, X_1)$ as having objects *cones*, i.e. triples (A, f_0, f_1) where A is a $\mathcal{C}$-object and $f_i : A \to X_i$ for $i = 0, 1$ are $\mathcal{C}$-arrows. We call A the *apex* and the f_i's the *legs*. Arrows $(A, f_0, f_1) \to (A', f_0', f_1')$ are simply $\mathcal{C}$-arrows $\varphi : A \to A'$ for which the following diagram commutes:

$$\begin{tikzcd}
A \arrow[rrr, "\varphi"] \arrow[dr, "f_0"'] \arrow[drr, "f_1"] & & & A' \arrow[dll, "f_0'"'] \arrow[dl, "f_1'"] \\
 & X_0 & X_1 &
\end{tikzcd}$$

The triple $(X_0 \times X_1, \pi_0, \pi_1)$ is then the terminal object in $\mathcal{C}/(X_0, X_1)$. This merely reformulates the diagram we used to define products since there $(S, \varphi_0, \varphi_1)$ was a cone and we stipulated that it had a unique cone arrow to $(X_0 \times X_1, \pi_0, \pi_1)$—this is precisely what it meant to be a terminal object! Since they are terminal objects, we automatically have that products are unique up to isomorphism—where this isomorphism is itself the unique one that consists of arrows that commute with projections. The cone constructions also allows us to codify the representability perspective of the product. More precisely, let $\mathrm{Cone}_{(X_0,X_1)} : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ map a $\mathcal{C}$-object A to the set of cones, i.e. objects of $\mathcal{C}/(X_0, X_1)$, with apex A. Furthermore, it maps a $\mathcal{C}$-arrow $\varphi : A \to A'$ to the set map $(A', f_0, f_1) \mapsto (A, \varphi \mathbin{/\!\!/} f_0, \varphi \mathbin{/\!\!/} f_1)$. We then have that this functor is represented by the product $X_0 \times X_1$; i.e. we have the following natural isomorphism:

$$\mathrm{Cone}_{(X_0,X_1)} \cong \mathcal{C}(-, X_0 \times X_1)$$

This is the case, since, as our definition established: a cone over X_0, X_1 is naturally equivalent to an arrow to the product $X_0 \times X_1$.

We now consider the dual case: consider a set map $\psi : R + S \to X$, e.g. if R and S represent the set of points on two different islands—and hence their disjoint union $R + S$ represents the set of points across both islands, X the set of possible temperatures, and ψ a map assigning a point its temperature. Dually to the previous example, we can split this up into a pair $(\psi_R : R \to X, \psi_S : S \to X)$, corresponding to the temperature maps for each individual island. Just as before, this yields a natural bijection:

$$\mathbf{Set}(R + S, X) \cong \mathbf{Set}(R, X) \times \mathbf{Set}(S, X).$$

Just as composing with the projection maps allows for decomposing maps into a product, precomposing with so called *inclusion maps* allows for decomposing maps out of a disjoint union. More specifically the inclusion map $\iota_R : R \to R+S$ is given by the inert rule $r \mapsto r$, which merely changes the ambient setting within which the element is considered. Then we have that $\psi_R = \iota_R \mathbin{/\!\!/} \psi$, and similarly for S.

More generally, given a category $\mathcal{C}$ and two objects X_i for $i = 0, 1$, the *coproduct* or *sum* $X_0 + X_1$, if it exists, is merely the product in $\mathcal{C}^{\mathrm{op}}$. Thus it is an object equipped with inclusion arrows $\iota_i : X_i \to X_0 + X_1$ for $i = 0, 1$, such that, given another object S and a pair of arrows $\varphi_i : X_i \to S$ for $i = 0, 1$, there exists a unique arrow $\varphi : X_0 + X_1 \to S$ making the following diagram in $\mathcal{C}$ commute:

$$\begin{tikzcd}
& S & \\
& X_0 \times X_1 \arrow[u, dashed, "\varphi"'] & \\
X_0 \arrow[ruu, "\varphi_0"] \arrow[ru, "\iota_0"'] & & X_1 \arrow[luu, "\varphi_1"'] \arrow[lu, "\iota_1"]
\end{tikzcd}$$

The coproduct $X_0 + X_1$ then satisfies the natural bijection:

$$\mathcal{C}(X_0 + X_1, -) \cong \mathcal{C}(X_0, -) \times \mathcal{C}(X_1, -).$$

We can encode this even more tightly as an initial object or a representable functor. Letting X_0, X_1 be objects of $\mathcal{C}$, define the category $(X_0, X_1)/\mathcal{C}$ as having objects *cocones*, i.e. triples (A, f_0, f_1) where A is a $\mathcal{C}$-object and $f_i : X_i \to A$ for $i = 0, 1$ are $\mathcal{C}$-arrows. We still call A the apex (but, if you're a stickler for symmetry, feel free to call it the *nadir*) and the ι_i's the legs (or *arms*). Arrows $(A, f_0, f_1) \to (A', f_0', f_1')$ are simply $\mathcal{C}$-arrows $\varphi : A \to A'$ for which the following diagram commutes:

$$\begin{tikzcd}
A \arrow[rrr, "\varphi"] & & & A' \\
& X_0 \arrow[ul, "f_0"] \arrow[urr, "f_0'"] & X_1 \arrow[ull, "f_1"] \arrow[ur, "f_1'"'] &
\end{tikzcd}$$

The triple $(X_0 + X_1, \iota_0, \iota_1)$ is then the initial object in $(X_0, X_1)/\mathcal{C}$. This merely reformulates the diagram we used to define coproducts since there $(S, \varphi_0, \varphi_1)$ was a cocone and we stipulated that it had a unique cocone arrow from $(X_0 + X_1, \iota_0, \iota_1)$—this is precisely what it meant to be an initial object! Again, we automatically have that coproducts are unique up to canonical isomorphism. The cocone constructions also allows us to codify the representability perspective of the coproduct. More precisely, let $\text{Cocone}_{(X_0,X_1)} : \mathcal{C} \to \textbf{Set}$ map a $\mathcal{C}$-object A to the set of cocones, i.e. objects of $(X_0, X_1)/\mathcal{C}$, with apex A. Furthermore, it maps a $\mathcal{C}$-arrow $\varphi : A \to A'$ to the set map $(A, f_0, f_1) \mapsto (A', f_0 \mathbin{/\!\!/} \varphi, f_1 \mathbin{/\!\!/} \varphi)$. We then have that this functor is represented by the coproduct $X_0 + X_1$; i.e. we have the following isomorphism:

$$\text{Cocone}_{(X_0,X_1)} \cong \mathcal{C}(X_0 + X_1, -)$$

This is the case, since, as our definition established: a cocone under X_0, X_1 is naturally equivalent to an arrow from the coproduct $X_0 + X_1$.

We note that, in the context of a thin category, i.e. preorder, $\overline{P}$ we had that the meet $x \wedge y$ satisfied, in arrow notation:

$$t \to x \wedge y \Leftrightarrow t \to x, t \to y,$$

I.e. we have the natural bijection

$$\overline{P}(t, x \wedge y) \cong \overline{P}(t, x) \times \overline{P}(t, y)$$

Hence the meet is a just a product, and, by duality, a join is just a coproduct.

Now, zooming out, let's consider products and coproducts in **Pre**. Recall that **Pre** inherited its initial and terminal object from **Set** for deep reasons we put off until the next section. For the same reason, this will also turn out to hold for products and coproducts! Our work isn't necessarily finished, though: it may be the case, as we'll show, that the underlying set of the product of two preorders is the Cartesian product of their underlying sets, but we still need to describe what the ordering is on this set—and the same for coproducts. This wasn't necessary before, simply because the initial object $\varnothing$ and terminal object $\star$ had a unique ordering. We start with the coproduct because it is more simple. More precisely, given preorders $(P_i, \preceq_i)$ for $i = 0, 1$, we define their coproduct $(P_0 + P_1, \preceq)$ by

$$\preceq (p, q) = \begin{cases} \preceq_i (p, q) & p, q \in P_i \\ \perp & \text{otherwise} \end{cases}$$

Said otherwise, if p and q belong to the same summand, then we merely relate them as we did before; otherwise, we do not relate them. This automatically makes the set inclusion maps $\iota_i : P_i \to P_0 + P_1$ monotonic since $p \preceq_i q \Rightarrow \iota_i(p) \preceq \iota_i(q)$. Furthermore, given a pair of monotone maps $\varphi_i : (P_i, \preceq_i) \to (Q, \sqsubseteq)$, this automatically assembled into a monotone map

$$\varphi_0 + \varphi_1 : (P_0 + P_1, \preceq) \to (Q, \sqsubseteq)$$

defined by $p \mapsto \varphi_i(p)$ for $p \in P_i$. This is automatically monotone since we needn't concern ourselves with preserving cross-summand relations! This vacuousness of satisfying property is at the heart of a universal construction. We now define the product to satisfy the dual condition: i.e. component maps should be vacuously combinable into a monotone map to the product. Towards this end, given preorders $(P_i, \preceq_i)$ for $i = 0, 1$, we define their product $(P_0 \times P_1, \preceq)$ as

$$\preceq((p_0, q_0), (p_1, q_1)) = \begin{cases} \top & [p_0 \preceq_0 q_0] \wedge [p_1 \preceq_1 q_1] \\ \bot & \text{otherwise} \end{cases}$$

Said otherwise, two ordered pairs (p_i, q_i) are related if and only if both of their respective components had been related. This allows projections to be vacuously monotone since $(p_0, p_1) \preceq (q_0, q_1) \Rightarrow p_i \preceq_i q_i$; and, furthermore, given monotone maps $\varphi_i : (Q, \sqsubseteq) \to (P_i, \preceq_i)$, this automatically assembles into a monotone map

$$\varphi_0 \times \varphi_1 : (Q, \sqsubseteq) \to (P_0 \times P_1, \preceq) :: q \mapsto (\varphi_0 q, \varphi_1 q)$$

since $q \sqsubseteq q' \Rightarrow \forall i : \varphi_i(q) \preceq_i \varphi_i(q') \Rightarrow \varphi(q) \preceq \varphi(q')$.

We now use these constructions to investigate the products and coproducts in the subcategories **Pos** and **Tot**. Before exploring them, we note that, if these categories have products or coproducts, then these must be the same as those constructed in **Pre**. This property is often referred to as *reflection* of products and coproducts, and is due to the following result.

Proposition 3.4. *Fully faithful functors reflect products. Formally, let $F : \mathcal{C} \to \mathcal{D}$ be a fully faithful functor, d_0, d_1 be $\mathcal{D}$-objects with product $d_0 \times d_1$, and c_0, c_1 be $\mathcal{C}$-objects for which $Fc_i = d_i$. Then, if $c_0 \times c_1$ exists, $F(c_0 \times c_1) = d_0 \times d_1$.*

Proof. Let $c \in |\mathcal{C}|$. Then consider the sequence of isomorphisms

$$\begin{aligned} \mathcal{D}(Fc, d_0 \times d_1) &\cong \mathcal{D}(Fc, d_0) \times \mathcal{D}(Fc, d_1) \\ &\cong \mathcal{C}(c, c_0) \times \mathcal{C}(c, c_1) \\ &\cong \mathcal{C}(c, c_0 \times c_1) \\ &\cong \mathcal{D}(Fc, F(c_0 \times c_1)) \end{aligned}$$

Since this did not rely on c, this gives the isomorphism of representable functors:

$$\mathcal{D}(-, d_0 \times d_1) \cong \mathcal{D}(-, F(c_0 \times c_1))$$

The full-faithfuness of the Yoneda embedding then implies the desired isomorphism

$$F(c_0 \times c_1) \cong d_0 \times d_1. \qquad \square$$

Since full-faithfulness does not rely on arrow orientation, duality gives us that fully faithful functors also reflect coproducts. Thus, to study products and coproducts in **Pos** and **Tot**, it suffices to check that the products and coproducts defined in **Pre** preserve posetality and totality of their components.

In the case of **Pos** both products and coproducts defined in **Pre** turn out to be products and coproducts in **Pos**. To see this, we check that if $(P_i, \preceq_i)$ are both posetal, then their product and coproduct is also posetal. Recall that posetality merely requires the antisymmetry condition: $x \preceq y, y \preceq x \Leftrightarrow x = y$. Since coproducts do not add any cross-summand relations, we receive no candidate pairs $p_i \in P_i$ for which $p_0 \preceq p_1$ or vice versa, and hence we automatically have that the preorder coproduct is also a poset. Now consider the product $(P_0 \times P_1, \preceq)$ and suppose

$(p_0, p_1) \preceq (q_0, q_1)$ and $(q_0, q_1) \preceq (p_0, p_1)$ and hence that $p_i \preceq_i q_i$ and $q_i \preceq_i p_i$, thus, by the posetality of the factors, forcing $p_i = q_i$. Therefore **Pos** inherits both its product and coproduct from **Pre**.

It is easy to see, in contrast, that the same doesn't hold for **Tot**. Since total orders require that for any pair p, q either $p \preceq q$ or $q \preceq p$, the coproduct of non-empty total orders is never a total order since we have no relations across summands. The same goes for products, and this is perhaps best exemplified with the example of $(\mathbb{Z} \times \mathbb{Z}, \leq)$—there's no way to compare $(3, 5)$ and $(5, 3)$, and hence products of total orders are typically not total.

Just as it did its terminal object, the category **Mon** inherits products from **Set**. More precisely, given monoids $(M_i, \odot_i, e_i)$ for $i = 0, 1$, we define their product as $(M_0 \times M_1, \odot, (e_0, e_1))$, where $\odot$ is defined component-wise:

$$(m_0, m_1) \odot (n_0, n_1) = (m_0 \odot_0 n_0, m_1 \odot_1 n_1).$$

It is a straightforward exercise to check that this satisfies the monoid axioms. The projection maps preserve the operation since

$$\pi_i((m_0, m_1) \odot (n_0, n_1)) = \pi_i(m_0 \odot_0 n_0, m_1 \odot_1 n_1) = m_i \odot_i n_i.$$

Similarly, two monoid homomorphisms $\varphi_i : N \to M_i$ vacuously assemble into a monoid homormorphism φ defined by $n \mapsto (\varphi_0 n, \varphi_1 n)$ since

$$\begin{aligned} \varphi(n \odot n') &= (\varphi_0(n \odot n'), \varphi_1(n \odot n')) \\ &= (\varphi_0 n \odot_0 \varphi_0 n', \varphi_1 n \odot_1 \varphi_1 n') \\ &= (\varphi_0 n, \varphi_1 n) \odot (\varphi_0 n', \varphi_1 n') \\ &= \varphi(n) \odot \varphi(n'). \end{aligned}$$

The analogous identity can be checked for unit elements. The coproduct of monoids, however, looks different from what we've seen thus far, but is related to the free monoid construction. More explicitly, given monoids $(M_i, \odot_i, e_i)$, we define their *free product* $(M_0 * M_1, \odot, e)$ as follows. Let $M_0 * M_1$ be the set of all words of alternating M_0 and M_1 non-unit elements with $\varnothing$ the empty word. Using superscripts to denote which monoid each element comes from, what follows are some $M_0 * M_1$ elements.

$$x^0 y^1 z^0, x^1 y^0 x^1 z^0, \varnothing$$

We note that these can start or end with a term from either M_0 or M_1, can include repeated elements, and can be of any finite length with the unique zero length word serving as the unit element. Since the unit element of any monoid is unique, we have to fuse the two monoids' units into a single unit. Since this is the case, we had to exclude units from our words since these could be absorbed into the element to their right or left. We will later see a more elegant characterization of this set.

Finally, we define the product $w \odot w'$ of two words w, w' as follows. If the final term of w and the first term of w' come from distinct monoids, then this product is merely the concatenation of the two words. If, in contrast, these two terms come from the same monoid, then we first concatenate the two lists, but then, so as to preserve alternation, replace these two adjacent elements with their product in the respective monoid to which they both belong. Towards greater formality, let $h(w)$ and $t(w)$ denote the first and last letter of w, respectively, and let $\overline{h}(w)$ and $\overline{t}(w)$ respectively denote *all but* the first and last term of w. Letting ; denote concatenation, we have $w = h(w); \overline{h}(w) = \overline{t}(w); t(w)$, and hence define $\odot$ by

$$w \odot w' = \begin{cases} \overline{t}(w); t(w) \odot_i h(w'); \overline{h}(w') & t(w), h(w') \in M_i \\ w; w' & \text{otherwise} \end{cases}$$

Note that, in the case of singleton words coming from the same monoid, this is merely their product in that monoid. We now define the inclusion maps $\iota_i : M_i \to M$ as sending non-unit elements to their corresponding singleton words, and sending the unit element to the empty word. These are homomorphisms since we do not alter the elements, merely recast them as words. Finally, given two monoid homomorphisms $\varphi_i : M_i \to N$, we show that these vacuously assemble into a monoid homomorphism φ defined term-wise. More precisely, $\varphi(a; b) = \varphi(a); \varphi(b)$, where, so as to respect the inclusion maps, we define $\varphi(a) = \varphi_i(a)$ for $a \in M_i$. This map automatically preserves products $w \odot w'$ when these equal $w; w'$. In the other case, i.e. when $t(w), h(w') \in M_i$, we have:

$$\begin{aligned} \varphi(w \odot w') &= \varphi(\overline{t}(w); t(w) \odot_i h(w'); \overline{h}(w')) \\ &= \varphi\overline{t}(w); \varphi_i[t(w) \odot_i h(w')]; \varphi\overline{h}(w') \\ &= \varphi\overline{t}(w); \varphi_i t(w) \odot_i \varphi_i h(w'); \varphi\overline{h}(w') \\ &= \varphi\overline{t}(w); \varphi t(w) \odot_i \varphi h(w'); \varphi\overline{h}(w') \\ &= \varphi(\overline{t}(w); t(w)) \odot \varphi(h(w')\overline{h}(w)) \\ &= \varphi w \odot \varphi w' \end{aligned}$$

Finally, we define $\varphi\varnothing = e_N$. The reader can check that this plays well with the computation above.

The product and coproduct of vector spaces is particularly interesting, in that it illuminates much of the underlying theory of linear algebra. It turns out, that in $\mathbf{Vect}_k$, the product and coproduct are both given by the direct sum $\oplus$, where the direct sum of two k-vector spaces V, W is defined as having vectors:

$$V \oplus W = \{(v, w) \mid v \in V, w \in W\},$$

with addition and scalar multiplication given component-wise:

$$c \cdot (v, w) = (c \cdot v, c \cdot w) \qquad (v, w) + (v', w') = (v + v', w + w').$$

When the product and coproduct in $\mathcal{C}$ are isomorphic, we call it a *biproduct*. The fact that the direct sum is a product follows from an identical argument as the monoid product: in both cases, we have component-wise operations. We hence leave this as an exercise to the reader. The fact that the direct sum is also a coproduct, however, is more interesting, and is made possible by the existence of 0 and $+$. More precisely, given V_i for $i = 0, 1$, we need inclusion maps $\iota_i : V_i \to V$. These are given by $v_0 \mapsto (v_0, 0)$ and $v_1 \mapsto (0, v_1)$. For sets, we did not have a canonical arrow into a Cartesian product from a factor—but because vector spaces are equipped with the special element 0, we can just define the other component of an included factor as 0. Similarly, we now need a way to assemble two maps $\varphi_i : V_i \to W$. We now wish to define the combined map $\varphi : V_0 \oplus V_1 \to W$ so as to satisfy, for the sake of a bijective correspondence, $\iota_i \mathbin{/\!\!/} \varphi = \varphi_i$. We do so by exploiting linearity:

$$\varphi(v_0, v_1) = \varphi(v_0, 0) + \varphi(0, v_1) = \varphi_0(v_0) + \varphi_1(v_1).$$

This cannot be done in the case of typical Cartesian products of sets since there is no way to break up a generic set map $f(u,v)$ into its u and v domain influences. Linearity, however, forces these to be independent!

If $\mathcal{C}$ has products and coproducts, then if we have an arrow $f : X_0 + X_1 \to Y_0 \times Y_1$, then this can be studied via an array of 4 component maps $f_i^j : X_i \to Y_j$ for $i, j = 0, 1$. When $\mathcal{C}$ has biproducts, this then becomes true for maps $\varphi : X_0 \oplus X_1 \to Y_0 \oplus Y_1$. In the context of vector spaces, this is precisely the idea of a block matrix! More precisely, for a linear map

$$L : V_0 \oplus V_1 \to W_0 \oplus W_1$$

we have a 2×2 array, or *block matrix* whose ij^{th} entry is a linear map $V_j \to V_i$. In addition to having biproducts, The category of k-vector spaces enjoys even more structure: all of its objects are isomorphic to *bipowers*, i.e. repeated biproducts, of isomorphic copies of a single object: the base field k. Recall that, for any vector space V, there exists a tuple $\mathcal{B} = (b_i)_{i=1}^n \subset V$, called a *basis*, such that, for any vector $v \in V$, there are unique $c_i \in k$ for which $v = \sum_i c_i b_i$. This can be recast as a direct sum as follows. Given a vector $v \in V$, let $kv = \{cv \mid c \in k\}$ be the one dimensional vector space of scalar multiples of v. Note that $kv \cong k$ via $v \mapsto 1$. Then, given a basis $\mathcal{B} = (b_i)_{i=1}^n \subset V$, we have the following sequence of isomorphisms, instantiating the direct sum decomposotion of V.

$$V \cong \bigoplus_{i=1}^{n} kb_i \cong k^n$$

The isomorphism ${}_{\mathcal{B}}[-]$ and its inverse $[-]_{\mathcal{B}}$ is given by the following composition of maps. Given a vector $v \in V$, we can uniquely write $v = \sum_i c_i b_i$. We then map v to the tuple $(c_1 b_1, \dots, c_n b_n)$ and then to $(c_1, \dots, c_n)$. In turn, given a tuple $(c_1, \dots, c_n) \in k^n$ to the tuple $(c_1 b_1, \dots, c_n b_n)$ and then to the sum $\sum_i c_i b_i$. The existence and uniqueness of a linear expression in terms of the basis for any vector is precisely what allows these maps to be mutual inverses. Now, given *any* linear map $L : V \to W$, and a basis $\mathcal{A}$ of V and $\mathcal{B}$ of W, we have the following composition:

$$\begin{tikzcd} V \arrow[r, "L"] & W \arrow[d, "{}_{\mathcal{B}}[-]"] \\ k^m \arrow[u, "{[-]_{\mathcal{A}}}"] \arrow[r, dotted, "{}_{\mathcal{B}}[L]_{\mathcal{A}}"'] & k^n \end{tikzcd}$$

Note that we used the shorthand ${}_{\mathcal{B}}[L]_{\mathcal{A}} = [-]_{\mathcal{A}} \mathbin{/\!\!/} L \mathbin{/\!\!/} {}_{\mathcal{B}}[-]$. The map ${}_{\mathcal{B}}[L]_{\mathcal{A}}$ is the matrix corresponding to L written in terms of the domain basis $\mathcal{A}$ and codomain basis $\mathcal{B}$. Recall that the ij^{th} entry ${}_i[L]_j$ of this matrix is a linear map $k \to k$, which is just a scalar; more precisely, it is the coefficient of the codomain basis b_i of the image of the domain basis a_j under the linear map L. In diagrammatic form, this entry can be encoded as follows.

$$\begin{tikzcd} V \arrow[r, "L"] & W \arrow[d] \\ ka_j \arrow[u] \arrow[r, dotted, "{}_i[L]_j"'] & kb_i \end{tikzcd}$$

Note that we chose the convention that the domain basis goes on the right so as to match the classic matrix composition ordering:

$$_{\mathcal{B}_2}[L]_{\mathcal{B}_1}\,_{\mathcal{B}_1}[L']_{\mathcal{B}_0} = {}_{\mathcal{B}_2}[LL']_{\mathcal{B}_0}.$$

Products and coproducts in $\mathcal{C}$ not only act on objects, but, when they exist, can be seen as functors $\mathcal{C} \times \mathcal{C} \to \mathcal{C}$. More precisely, we map $\mathcal{C} \times \mathcal{C}$-objects—i.e. pairs (X, Y)—respectively to $X \times Y$ and $X + Y$, and $\mathcal{C} \times \mathcal{C}$-arrows—i.e. pairs $(f, g) : (X, Y) \to (X', Y')$—respectively to arrows $X \times Y \to X' \times Y'$ and $X + Y \to X' + Y'$. We define the product arrow in terms of universal properties as follows.

$$\begin{tikzcd}
 & X \arrow[r, "f"] & X' \\
X \times Y \arrow[ur, "\pi_X"] \arrow[dr, "\pi_Y"'] \arrow[r, dashed] & X' \times Y' \arrow[ur, "\pi_{X'}"'] \arrow[dr, "\pi_{Y'}"] & \\
 & Y \arrow[r, "g"'] & Y'
\end{tikzcd}$$

The case of coproduct is merely dual to the case of product above and is left as an exercise. In the case of **Set**, these are defined in the intuitive way: $(x, y) \mapsto (fx, gy)$ and $x \mapsto fx, g \mapsto gy$ respectively. In the case of biproducts and a zero object (a simultaneously initial and terminal object)—as is the situation in $\mathbf{Vect}_k$—this can be made even simpler. Since we have a zero object, we can define the *zero morphism* $X \to Y$ for *any* pair X, Y as the following composition:

$$\begin{tikzcd}
X \arrow[r, dashed] & 0 \arrow[r, dashed] & Y
\end{tikzcd}$$

Then, since any arrow $X \oplus Y \to X' \oplus Y'$ is merely an arrow for each choice of domain and codomain summand, simply map $(f, g) : (X, Y) \to (X', Y')$ to the arrow $X \oplus Y \to X' \oplus Y'$ defined by the components as follows:

$$\begin{bmatrix} f : X \to X' & 0 \\ 0 & g : Y \to Y' \end{bmatrix}$$

Thus the direct sum of linear maps $\oplus_i^n L_i : V_i \to W_i$ is the map $L : \oplus_i V_i \to \oplus_i W_i$ given by the following *block diagonal* matrix.

$$\begin{bmatrix} L_1 & 0 & \dots & 0 \\ 0 & L_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & L_n \end{bmatrix}$$

Thus, we can define a diagonal matrix as a direct sum of scalar maps $\oplus_i(c_i : k \to k)$.

Note that, although the direct sum of vector spaces involves the product of their unerlying sets, its dimension is the sum of their respective dimensions—more precisely, the basis of the direct sum is the disjoint union of the bases of each summand. This is true for a deep reason to be explained in the next section. What about a vector space whose basis is the product of the bases of some pair of vector spaces? It turns out that there is a universal construction that achieves this. To motivate this construction, let V and V' be vector spaces with bases $\mathcal{A} = (a_j)_{j \in J}$

and $\mathcal{A}' = (a_i)_{i \in I}$. Let L be a linear map, with domain the *set* of vectors $V \times V'$, and R a *bilinear* map with domain $V \times V'$. We then have the following scenarios.

$$\begin{aligned} L(v,w) &= L(\textstyle\sum_j c_j a_j, \sum_i c_i a_i) \\ &= L(\textstyle\sum_j c_j a_j, 0) + L(0, \sum_i c_i a_i) \\ &= \textstyle\sum_j c_j L(a_j, 0) + \sum_i c_i L(0, a_i) \\ &= \textstyle\sum_{k \in I+J} c_k L \hat{a}_k \end{aligned}$$

where we let $\hat{a}_k$ be $(a_j, 0)$ for $k \in J$ and $(0, a_i)$ for $k \in I$. This notation is made to transparently imply the idea that a linear map of two arguments is determined by the disjoint union of the bases for these two arguments' spaces. Now consider the case of bilinearity.

$$\begin{aligned} R(v,w) &= R(\textstyle\sum_j c_j a_j, \sum_i c_i a_i) \\ &= \textstyle\sum_i \sum_j c_j c_i R(a_j, a_i) \\ &= \textstyle\sum_{(i,j) \in I \times J} c_i c_j R(a_j \otimes a_i) \end{aligned}$$

where the *tensor* $a_j \otimes a_i$ is a formal symbol that represents the ordered pair (a_j, a_i), but now conceived as a generic element of a new domain vector space that we define in order to reconceptualize R as a linear map instead of a bilinear one. Note that R is determined by pairs of the two arguments, and hence will have as basis the Cartesian product of the bases of its factors, or *tensorands*. The tensor product will thus be multiplicative in dimension. We may construct the *tensor product* $V \otimes V'$ explicitly as follows.

$$V \otimes V' = \{\textstyle\sum_{i,j} c_{ij}(v_i \otimes v'_j) \mid v_i \in V, v'_j \in V', c_{ij} \in k\}/\sim,$$

where the equivalence relation encodes the multilinearity that we wish to be preserved by an outgoing linear map. More precisely:

$$[\textstyle\sum_i c_i v_i] \otimes [\sum_j c'_j v'_j] \sim \sum_{i,j} c_i c'_j (v_i \otimes v_j).$$

A linear map $\tilde{R} : V \otimes V' \to W$ then behaves like a bilinear map $R : V \times V' \to W$:

$$\begin{aligned} \tilde{R}([\textstyle\sum_i c_i v_i] \otimes [\sum_j c'_j v'_j]) &= \tilde{R}(\textstyle\sum_{i,j} c_i c'_j (v_i \otimes v_j)) \\ &= \textstyle\sum_{i,j} c_i c'_j \tilde{R}(v_i \otimes v_j) \end{aligned}$$

This can be made more precise via a universal property:

$$\begin{tikzcd} V \times V' \arrow[r, "R"] \arrow[d, "\rho"'] & W \\ V \otimes V' \arrow[ur, dashed, "\tilde{R}"'] & \end{tikzcd}$$

The map ρ is given by the rule $(v, v') \mapsto v \otimes v'$, and the dashed line, as usual, represents the unique arrow—in $\mathbf{Vect}_k$—that makes the diagram commute. Note that the non-dashed arrows are in this case *not* arrows in $\mathbf{Vect}_k$ but rather are bilinear maps. As usual, this universal property can be recast as a representable functor. More precisely let $\mathrm{Bilin}(V \times V', -) : \mathbf{Vect}_k \to \mathbf{Set}$ map a vector space W to the set of bilinear maps $V \times V' \to W$. Then the universal property gives the following natural isomorphism:

$$\mathrm{Bilin}(V \times V', -) \cong \mathbf{Vect}_k(V \otimes V', -).$$

Thus, although the tensor product is neither product nor coproduct in $\mathbf{Vect}_k$, it is nonetheless a universal binary operation on vector spaces, and one that plays the role of multiplication—at the least at the level of dimensionality—relative to the direct sum's role of addition. We will explore this further in the subsequent section.

Another pair of universal constructions of note are so called *equalizers* and their dual *coequalizers*. Both constructions begin with a pair of parallel arrows $f, g : X \to Y$, and ask the following dual questions. The equalizer $\mathrm{eq}(f, g)$ is an arrow $\varphi : A \to X$ for which $\varphi \mathbin{/\!\!/} f = \varphi \mathbin{/\!\!/} g$, such that any other arrow $\beta : B \to X$ with the same property must be a composition $\beta = \alpha \mathbin{/\!\!/} \varphi$ for some $\alpha : B \to A$. This can be expressed as a universal diagram as follows:

$$\begin{tikzcd}
A \arrow[r, "\varphi"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \\
B \arrow[u, dashed, "\alpha"] \arrow[ur, "\beta"'] & &
\end{tikzcd}$$

The coequalizer $\mathrm{coeq}(f, g)$ is given by the dual condition:

$$\begin{tikzcd}
X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \arrow[r, "\psi"] \arrow[dr, "\omega"'] & V \arrow[d, dashed, "\nu"] \\
& & W
\end{tikzcd}$$

In the case of **Set**, we will show that the equalizer and coequalizer—using the same variable names as in the diagrams above—can be given via the following formula.

$$A = \{x \in X \mid fx = gx\}$$
$$V = Y/\sim \text{ where } y \sim y' \Leftrightarrow [y = fx, y' = gx]$$

By construction, these satisfy the commutativity properties. We now prove universality. We first do the equalizer case. Suppose there is some $\beta : B \to X$ for which $\beta \mathbin{/\!\!/} f = \beta \mathbin{/\!\!/} g$, i.e. $f(\beta b) = g(\beta b)$ for every $b \in B$. This implies that $\beta b \in A$, and hence that the image of β is a subset of A. Thus we may simply define α by the rule given by β, except with codomain A. Now consider the coequalizer case; i.e. that there is a map $\omega : Y \to W$ for which $f \mathbin{/\!\!/} \omega = g \mathbin{/\!\!/} \omega$, i.e. for which $\omega(fx) = \omega(gx)$ for all $x \in X$. But then this means that ω can be defined on equivalence classes $[y]$ based on the equivalence relation $\sim$, and hence can be seen as a two step process: first via applying ψ, which quotients by $\sim$, and then by applying ω to the equivalence relation.

In the context of **Pre**, the equalizer and coequalizer have underlying set given by the set (co)equalizer. The ordering structure on the equalizer A is made to maintain that it is a sub-order of X—adding any further relations would cause φ to fail to be monotone, and deleting any relations would create an obstruction for maps from B to factor through A. In the case of the coequalizer, we must now ask what it means for $[y] \preceq [y']$. Since ψ is monotone, we *must* always have that, when $y \preceq y'$, that $[y] \preceq [y']$. To make this relation well defined under the equivalence class, we define the ordering as follows.

$$\preceq([y], [y']) = \begin{cases} \top & \exists z, z' : y \sim z \preceq z' \sim y' \\ \bot & \text{otherwise} \end{cases}$$

As was the case for products and coproducts, **Pos** and **Tot** inherit their equalizers from **Pre**—it is easy to see that posets are closed under taking both equalizers and coequalizers, and that total orders are closed under taking equalizers but typically not under coequalizers.

We will not formally discuss (co)equalizers in the context of **Mon**. The idea would involve defining things in terms of generators so that the relevant arrows become homomorphisms. In the context of $\mathbf{Vect}_k$, given a linear map $L : V \to W$, we define its *kernel* $\ker L$ and *cokernel* $\operatorname{coker} L$ respectively as be the equalizer and coequalizer of the pair $L, 0 : V \to W$. Set theoretically, we define these as follows.

$$\begin{aligned} \ker L &= \{v \in V \mid Lv = 0\} \\ \operatorname{coker} L &= W/\sim \ \text{ where } w \sim w' \Leftrightarrow \exists v : w - w' = Lv \end{aligned}$$

Then, it is easy to check that the equalizer and coequalizer of $L, L' : V \to W$ are respectively $\ker(L - L')$ and $\operatorname{coker}(L - L')$.

We now introduce a highly general dual pair of universal constructions. Let $\mathcal{C}$ be a category, serving as a setting, Δ a small category, serving as a shape, and $\tau : \Delta \to \mathcal{C}$ be a $\mathcal{C}$-diagram of shape Δ. Define a *cone* $(c, (f_d)_{d \in |\Delta|})$ over τ as having the data of a $\mathcal{C}$-object c and a collection of arrows $f_d : c \to \tau(d)$ for all Δ-objects d. Furthermore, this data must satisfy the condition that for all Δ-arrows $i : d \to d'$, the following diagram commutes.

$$\begin{tikzcd}
c \arrow[d, "f_d"'] \arrow[dr, "f_{d'}"] & \\
\tau(d) \arrow[r, "\tau(i)"'] & \tau(d')
\end{tikzcd}$$

Dually, a *cocone* $(c, (f_d)_{d \in |\Delta|})$ under τ has an equivalent definition with the exception that the arrows $f_d : \tau(d) \to c$ go the other way. In a context where Δ has a visually tractable form, we often represent cones and cocones via a single picture. For example, let Δ be the preorder defined by the following graph.

$$\begin{tikzcd}
d_1 \arrow[r] & d_2 & d_3 \\
& d_4 \arrow[ul] \arrow[u] \arrow[ur] & d_5 \arrow[u]
\end{tikzcd}$$

Given the diagram $\tau : \Delta \to \mathcal{C}$, a cone over τ can then be depicted as the following commutative diagram.

$$\begin{tikzcd}
c \arrow[d] \arrow[dr] \arrow[drr] \arrow[ddr] \arrow[ddrr] & & \\
\tau(d_1) \arrow[r] & \tau(d_2) & \tau(d_3) \\
& \tau(d_4) \arrow[ul] \arrow[u] \arrow[ur] & \tau(d_5) \arrow[u]
\end{tikzcd}$$

By virtue of commutativity, a subset of the legs can determine the cone. In this context, the following diagram uniquely determines the one above.

$$\begin{tikzcd}
c \arrow[ddr] \arrow[ddrr] & & \\
\tau(d_1) \arrow[r] & \tau(d_2) & \tau(d_3) \\
& \tau(d_4) \arrow[ul] \arrow[u] \arrow[ur] & \tau(d_5) \arrow[u]
\end{tikzcd}$$

Now consider a cocone under τ.

$$\begin{tikzcd}
c & & \\
\tau(d_1) \arrow[u] \arrow[r] & \tau(d_2) \arrow[ul] & \tau(d_3) \arrow[ull] \\
& \tau(d_4) \arrow[ul] \arrow[u] \arrow[ur] \arrow[uul] & \tau(d_5) \arrow[u] \arrow[uull]
\end{tikzcd}$$

This diagram too can be simplified, albeit differently than in the cone case.

$$\begin{tikzcd}
c & & \\
\tau(d_1) \arrow[r] & \tau(d_2) \arrow[ul] & \tau(d_3) \arrow[ull] \\
& \tau(d_4) \arrow[ul] \arrow[u] \arrow[ur] & \tau(d_5) \arrow[u]
\end{tikzcd}$$

We now define a pair of notions that directly generalize products and coproducts. Let $\mathcal{C}/\tau$ be the category with objects cones over τ and arrows $(c,(f_d)) \to (c',(f'_d))$ $\mathcal{C}$-arrows $\varphi : A \to A'$ for which $f'_d = \varphi \mathbin{/\!\!/} f_d$. Then the *limit* of τ, denoted $\lim \tau$ is the terminal object of $\mathcal{C}/\varphi$. Dually, define the category $\tau/\mathcal{C}$ as having objects cocones under τ and arrows $(c,(f_d)) \to (c',(f'_d))$ $\mathcal{C}$-arrows $\varphi : A \to A'$ for which $f'_d = f_d \mathbin{/\!\!/} \varphi$. Then the *colimit* of τ, denoted $\operatorname{colim} \tau$, is the initial object of $\tau/\mathcal{C}$. Now let $\mathrm{Cone}_\tau : \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$ map a $\mathcal{C}$-object A to the set of cones over τ, i.e. $\mathcal{C}/\tau$-objects, with apex A. Dually, let $\mathrm{Cocone}_\tau : \mathcal{C} \to \mathbf{Set}$ map a $\mathcal{C}$-object A to the set of cocones under τ, i.e. $\tau/\mathcal{C}$-objects, with apex A. We then have the following isomorphisms.

$$\mathrm{Cone}_\tau \cong \mathcal{C}(-, \lim \tau)$$
$$\mathrm{Cocone}_\tau \cong \mathcal{C}(\operatorname{colim} \tau, -)$$

Let Δ be the discrete category on $\underline{2}$ and $\tau(i) = X_i$. We then have the following.

$$\begin{aligned} \mathcal{C}/\tau &= \mathcal{C}/(X_0, X_1) \\ \tau/\mathcal{C} &= (X_0, X_1)/\mathcal{C} \\ \mathrm{Cone}_\tau &= \mathrm{Cone}_{X_0,X_1} \\ \mathrm{Cocone}_\tau &= \mathrm{Cocone}_{X_0,X_1} \\ \lim \tau &= X_0 \times X_1 \\ \mathrm{colim}\, \tau &= X_0 + X_1 \end{aligned}$$

Now let Δ be given by the following quiver.

$$\begin{tikzcd} d \arrow[r, shift left, "0"] \arrow[r, shift right, "1"'] & d' \end{tikzcd}$$

Let τ be given by $0 \mapsto f, 1 \mapsto g$. Then $\lim \tau = \mathrm{eq}(f, g)$ and $\mathrm{colim}\, \tau = \mathrm{coeq}(f, g)$. In addition to these two common shapes, there are two more pairs that are arise sufficiently frequently that they have names. Denote respectively by $\to\leftarrow$ and $\leftarrow\to$ the following two diagrams.

$$\begin{tikzcd} d' \arrow[r, "0"] & d & d'' \arrow[l, "1"'] \end{tikzcd} \qquad \begin{tikzcd} d' & d \arrow[l, "0"'] \arrow[r, "1"] & d'' \end{tikzcd}$$

If $\tau : [\to\leftarrow] \to \mathcal{C}$, we call $\lim \tau$ a *pullback*, and if $\sigma : [\leftarrow\to] \to \mathcal{C}$, we call $\mathrm{colim}\, \sigma$ a *pushout*. The other two configurations—$\mathrm{colim}\, \tau$ and $\lim \sigma$—are seldom encountered. Letting τ and σ both be given by the rules $d' \mapsto X, d \mapsto Z, d'' \mapsto Y, 0 \mapsto f, 1 \mapsto g$, we often write $\lim \tau = X \times_Z Y$ and $\mathrm{colim}\, \sigma = X +_Z Y$, and often refer to them as *fibered* products and coproducts, respectively. In this context, we often depict pullbacks and pushouts as follows.

$$\begin{tikzcd} X \times_Z Y \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d, "f"] \\ Y \arrow[r, "g"'] & Z \end{tikzcd} \qquad \begin{tikzcd} Z \arrow[r, "f"] \arrow[d, "g"'] & X \arrow[d] \\ Y \arrow[r] & X +_Z Y \arrow[ul, phantom, "\ulcorner", very near start] \end{tikzcd}$$

The corner symbols are used to distinguish these two situations, both from each other and from a mere commutative square. We will only cover examples of pullbacks and pushouts in **Set**. In general, these are given by the following formulae.

$$\begin{aligned} X \times_Z Y &= (X \times Y)|_{fx=gy} \\ X +_Z Y &= (X + Y)/[fz \sim gz] \end{aligned}$$

This has some interesting special cases. For example, suppose A, B are both subsets of S and we consider the pullback of the inclusion maps ι_A, ι_B. Then

$$A \times_S B = A \times B|_{\iota_A a = \iota_B b} = \{(s, s) \mid s \in A \cap B\} \cong A \cap B.$$

Hence we have the set intersection. Now, with this intersection in hand, consider $A \cap B$ with inclusions ι'_A, ι'_B to A, B. Then the pushout is the classic union:

$$A +_{A \cap B} B = A + B/[\iota'_A z \sim \iota'_B z] = A \cup B.$$

We now consider a differently flavored example. Let $f : X \to Y$ be a map and $\iota : S \to Y$ a subset inclusion. Then the pullback is given by the preimage:

$$X \times_Y S = X \times S|_{fx = \iota s} = \{x \in X \mid fx \in S\} = f^*(S).$$

We end this section with a remark about limits and colimits of diagrams of shape given by the thin category $(\mathbb{N}, \leq)$, which we denote by ω, and depict as follows:

$$0 \to 1 \to 2 \to \cdots \to n \to \cdots$$

Classically, limits and colimits of diagrams of this shape have respectively been called *inverse limits* and *direct limits*. We will avoid this terminology since it flips the verbal associations of the general limit and colimit terminology.

Let $\tau : \omega \to \mathbf{Set}$ be given on objects by $n \mapsto \mathbf{n} = \{1, \ldots, n\}$ and on arrows as inclusion maps. We now show why $\operatorname{colim} \tau \cong \mathbb{N}$.

$$\begin{tikzcd}
\mathbf{0} \arrow[r] \arrow[d] \arrow[ddrr] & \mathbf{1} \arrow[r] \arrow[dl] \arrow[ddr] & \cdots \arrow[r] & \mathbf{n} \arrow[r] \arrow[dlll, "\iota_n"'] \arrow[ddl, "f_n"] & \cdots \\
\mathbb{N} \arrow[drr, dashed, "f"'] & & & & \\
& & A & &
\end{tikzcd}$$

The universal cocone legs are inclusions. We must now construct a map $f : \mathbb{N} \to A$. By defining this via the rule $n \mapsto f_n(n)$, we have that $\iota_n \mathbin{/\!\!/} f = f_n$. This is well defined by the cone condition, which forces $f_n(i) = f_m(i)$ whenever $i < m < n$.

We now consider a limit over a diagram of such a shape. Given some set X, let $\tau : \omega^{\mathrm{op}} \to \mathbf{Set}$ be given on objects by $n \mapsto X^n$ and on arrows as projections $\eta_n : X^n \to X^{n-1} :: (x_1, \ldots, x_{n-1}, x_n) \mapsto (x_1, \ldots, x_{n-1})$. We show that $\lim \tau \cong X^{\mathbb{N}}$.

$$\begin{tikzcd}
\star & X \arrow[l] & \cdots \arrow[l] & X^n \arrow[l] & \cdots \arrow[l] \\
X^{\mathbb{N}} \arrow[u] \arrow[ur] \arrow[urrr, "\Pi_n"] & & & & \\
& & A \arrow[uull] \arrow[uul] \arrow[uur, "f_n"'] \arrow[ull, dashed, "f"] & &
\end{tikzcd}$$

Define the universal cone leg Π_n by the map $s \mapsto s(n)$. A map $f : A \to X^{\mathbb{N}}$ is equivalent to a map $F : \mathbb{N} \times A \to X$. Let $\pi_n : X^n \to X$ be given by the projection onto the final factor. We then define $F(n, -)$ as the following composition.

$$\begin{tikzcd}
A \arrow[r, "f_n"] \arrow[dr, "{F(n,-)}"'] & X^n \arrow[d, "\pi_n"] \\
& X
\end{tikzcd}$$

It turns out that all limits are in some sense generated by products and equalizers; and, dually, all colimits by coproducts and coequalizers. We conclude this section by stating without proof a result that formalizes this intuition.

Proposition 3.5. *Let $\mathcal{C}$ be a category with products and equalizers. Then, given any diagram $\tau : \Delta \to \mathcal{C}$, $\lim \tau$ exists in $\mathcal{C}$.*