Chapter 4

Collaborative design: Profunctors, categorification, and monoidal categories

4.1 Can we build it?

When designing a large-scale system, many different fields of expertise are joined into a single project. Thus the whole project team is divided into multiple sub-teams, each of which is working on a sub-project. And we recurse downward: the sub-project is again factored into sub-sub-projects, each with their own team. One could refer to this sort of hierarchical design process as *collaborative design*, or co-design. In this chapter, we discuss a mathematical theory of co-design, due to Andrea Censi [**Censi:2015a**].

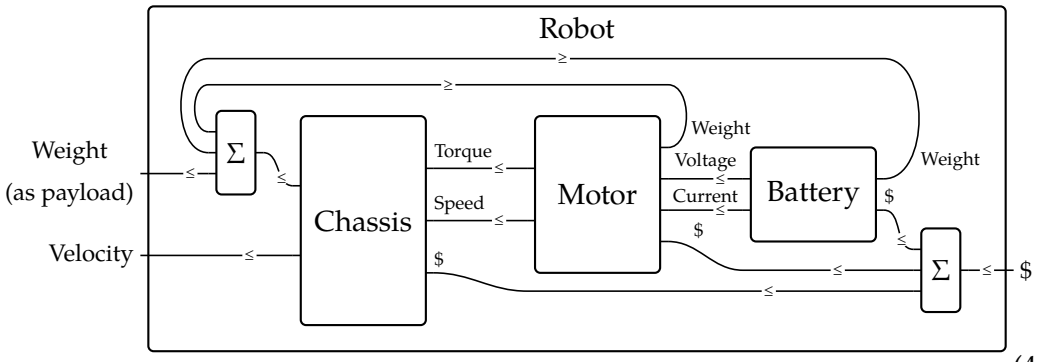
Consider just one level of this hierarchy: a project and a set of teams working on it. Each team is supposed to *provide* resources—sometimes called "functionalities"—to the project, but the team also *requires* resources in order to do so. Different design teams must be allowed to plan and work independently from one another in order for progress to be made. Yet the design decisions made by one group effect the design decisions others can make: if A wants more space in order to provide a better radio speaker, then B must use less space. So these teams—though ostensibly working independently—are dependent on each other after all.

The combination of dependence and independence is crucial for progress to be made, and yet it can cause major problems. When a team requires more resources than it originally expected to require, or if it cannot provide the resources that it originally claimed it could provide, the usual response is for the team to issue a design-change notice. But these effect neighboring teams: if team A now requires more than originally claimed, team B may have to change their design. Thus these design-change notices can ripple through the system through feedback loops and cause whole projects to fail [**Subrahmanian.Lee.Granger:2015a**].

As an example, consider the design problem of creating a robot to carry some load at some velocity. The top-level planner breaks the problem into three design teams: team chassis, team motor, and team battery. Each of these teams could break up into multiple parts and the process repeated, but let's remain at the top level and consider the resources produced and the resources required by each of our three teams.

The chassis in some sense provides all the functionality—it carries the load at the velocity—but it requires some things in order to do so. It requires money, of course, but more to the point it requires a source of torque and speed. These are supplied by the motor, which in turn needs voltage and current from the battery. Both the motor and the battery cost money, but more importantly they need to be carried by the chassis: they become part of the load. A feedback loop is created: the chassis must carry all the weight, even that of the parts that power the chassis. A heavier battery might provide more energy to power the chassis, but is the extra power worth the heavier load?

In the following picture, each part—chassis, motor, battery, and robot—is shown as a box with ports on the left and right. The functionalities, or resources produced by the part are on the left of the box, and the resources required by the part are on the right.



(4.1)

The boxes marked Σ correspond to summing inputs. These boxes are not to be designed, but we will see later that they fit easily into the same conceptual framework. Note also the ≤'s on each wire; they indicate that if box A requires a resource that box B produces, then A's requirement must be less-than-or-equal-to B's production.

To formalize this a bit more, let's call diagrams like the one above *co-design diagrams*. Each of the wires in a co-design diagram represents a poset of resources. For example, in Eq. (4.1) every wire corresponds to a resource type—weights, velocities, torques, speeds, costs, voltages, and currents—where resources of each type can be ordered from less useful to more useful. In general, these posets do not have to be linear orders, though in the above cases each will likely correspond to a linear order: \$10 ≤ \$20, 5W ≤ 6W, and so on.

Each of the boxes in a co-design diagram corresponds to what we call a *feasibility relation*. A feasibility relation matches resource production with requirements. For every pair $(p, r) \in P \times R$, where P is the poset of resources to be produced and R is the poset of resources to be required, the box says "true" or "false"—feasible or

infeasible—for that pair. In other words, "yes I can provide p given r" or "no, I cannot provide p given r".

Feasibility relations hence define a function $\Phi\colon P \times R \to \mathbf{Bool}$. For a function $\Phi\colon P \times R \to \mathbf{Bool}$ to make sense as a feasibility relation, however, there are two conditions:

(a) If $\Phi(p, r) = \texttt{true}$ and $p' \leq p$, then $\Phi(p', r) = \texttt{true}$.
(b) If $\Phi(p, r) = \texttt{true}$ and $r \leq r'$ then $\Phi(p, r') = \texttt{true}$.

These conditions, which we will see again in Definition 4.1, say that if you can produce p given resources r, you can (a) also produce less $p' \leq p$ with the same resources r, and (b) also produce p given more resources $r' \geq r$. We will see that these two conditions are formalized by requiring Φ to be a monotone map $P^{\mathrm{op}} \times R \to \mathbf{Bool}$.

A *co-design problem*, represented by a co-design diagram, asks us to find the composite of some feasibility relations. It asks, for example, given these capabilities of the chassis, motor, and battery teams, can we, together, build a robot? Indeed, a co-design diagram factors a problem—for example, that of designing a robot—into interconnected subproblems, as in Eq. (4.1). Once the feasibility relation is worked out for each of the subproblems, i.e. the inner boxes in the diagram, the mathematics provides an algorithm producing the feasibility relation of the whole outer box. This process can be recursed downward, from the largest problem to tiny subproblems.

In this chapter, we will understand co-design problems in terms of enriched profunctors, in particular **Bool**-profunctors. A **Bool**-profunctor is like a bridge connecting one poset to another. We will show how the co-design framework gives rise to a structure known as a compact closed category, and that any compact closed category can interpret the sorts of wiring diagrams we see in Eq. (4.1).

4.2 Enriched profunctors

In this section we will understand how co-design problems form a category. Along the way me will develop some abstract machinery that will allow us to replace poset design spaces with other enriched categories.

4.2.1 Feasibility relationships as Bool-profunctors

The theory of co-design is based on posets: each resource—e.g. velocity, torque, or \$—is structured as a poset. The order $x \leq y$ represents the *availability of x given y*, i.e. that whenever you have y, you also have x. For example, in our poset of wattage, if 5W $\leq$ 10W, it means that whenever we are provided 10W, we implicitly also have 5W.

We know from **??** that a poset $\mathcal{X}$ can be conceived of as a **Bool**-category. Given $x, y \in X$, we have $\mathcal{X}(x, y) \in \mathbb{B}$; this value responds to the assertion "x is available given y," marking it either `true` or `false`.

Our goal is to see feasibility relations as **Bool**-profunctors, which are a special case of something called enriched profunctors. Indeed, we hope that this chapter will give

you some intuition for profunctors, arising from the table

Bool-category	poset
Bool-functor	monotone map
Bool-profunctor	feasibility relation

Because enriched profunctors are a touch abstract, we first concretely discuss **Bool**-profunctors as feasibility relations. Recall that if $\mathcal{X} = (X, \leq)$ is a poset, then its opposite $\mathcal{X}^{\mathrm{op}} = (X, \geq)$ has $x \geq y$ iff $y \leq x$.

Definition 4.1. Let $\mathcal{X} = (X, \leq_X)$ and $\mathcal{Y} = (Y, \leq_Y)$ be posets. A *feasibility relation* for $\mathcal{X}$ given $\mathcal{Y}$ is a monotone map

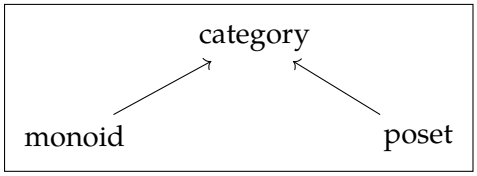
$$\Phi \colon \mathcal{X}^{\mathrm{op}} \times \mathcal{Y} \to \mathbf{Bool} \tag{4.2}$$

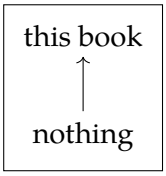
We denote this by $\Phi \colon \mathcal{X} \nrightarrow \mathcal{Y}$.

Given $x \in X$ and $y \in Y$, if $\Phi(x, y) = \texttt{true}$ we say *x can be obtained given y*.

As mentioned in the introduction, the requirement that Φ is monotone says that if $x' \leq_X x$ and $y \leq_Y y'$ then $\Phi(x, y) \leq_{\mathbf{Bool}} \Phi(x', y')$. In other words, if x can be obtained given y, and if x' is available given x, then x' can be obtained given y. And if furthermore y is available given y', then x' can also be obtained given y'.

Exercise 4.2. Suppose we have the posets

$\mathcal{X} :=$ 

$\mathcal{Y} :=$ 

1. Draw the Hasse diagram for the poset $\mathcal{X}^{\mathrm{op}} \times \mathcal{Y}$.
2. Write down a profunctor $\Lambda \colon \mathcal{X} \nrightarrow \mathcal{Y}$ and, reading $\Lambda(x, y) = \texttt{true}$ as "my uncle can explain x given y", give an interpretation of the fact that the preimage of $\texttt{true}$ forms an upper set in $\mathcal{X}^{\mathrm{op}} \times \mathcal{Y}$. ◊

To generalize the notion of feasibility relation, we must notice that the symmetric monoidal poset **Bool** has more structure than just that of a symmetric monoidal poset: as mentioned in **??**, **Bool** is a quantale. That means it has all joins $\vee$, and a closure operation, which we'll write $\Rightarrow \colon \mathbb{B} \times \mathbb{B} \to \mathbb{B}$. By definition, this operation satisfies the property that for all $b, c, d \in \mathbb{B}$ one has

$$b \wedge c \leq d \quad \text{iff} \quad b \leq (c \Rightarrow d). \tag{4.3}$$

The operation $\Rightarrow$ is given by the following table:

c	d	$c \Rightarrow d$
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`true`
`false`	`false`	`true`

(4.4)

Exercise 4.3. Show that $\Rightarrow$ as defined in Eq. (4.4) indeed satisfies Eq. (4.3). ◊

On an abstract level, it is the fact that **Bool** is a quantale which makes everything in this chapter work; any other (commutative) quantale also defines a way to interpret co-design diagrams. For example, we could use the quantale **Cost**, which would describe not *whether* x is available given y but the *cost* of obtaining x given y; see **????**.

4.2.2 $\mathcal{V}$-profunctors

We are now ready to recast Eq. (4.2) in abstract terms. Recall the notions of enriched product (**??**), enriched functor (**??**), and commutative quantale (**??**).

Definition 4.4. Let $\mathcal{V} = (V, \leq, I, \otimes)$ be a commutative quantale, and let $\mathcal{X}$ and $\mathcal{Y}$ be $\mathcal{V}$-categories. A *$\mathcal{V}$-profunctor* from $\mathcal{X}$ to $\mathcal{Y}$, denoted $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$, is a $\mathcal{V}$-functor

$$\Phi\colon \mathcal{X}^{\mathrm{op}} \times \mathcal{Y} \to \mathcal{V}.$$

Note that a $\mathcal{V}$-functor must have $\mathcal{V}$-categories for domain and codomain, so here we are considering $\mathcal{V}$ as enriched in itself; see **??**.

Exercise 4.5. Show that a $\mathcal{V}$-profunctor (Definition 4.4) is the same as a function $\Phi\colon \mathrm{Ob}(\mathcal{X}) \times \mathrm{Ob}(\mathcal{Y}) \to V$ such that for any $x, x' \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$ the following inequality holds in $\mathcal{V}$:

$$\mathcal{X}(x', x) \otimes \Phi(x, y) \otimes \mathcal{Y}(y, y') \leq \Phi(x', y').$$ ◊

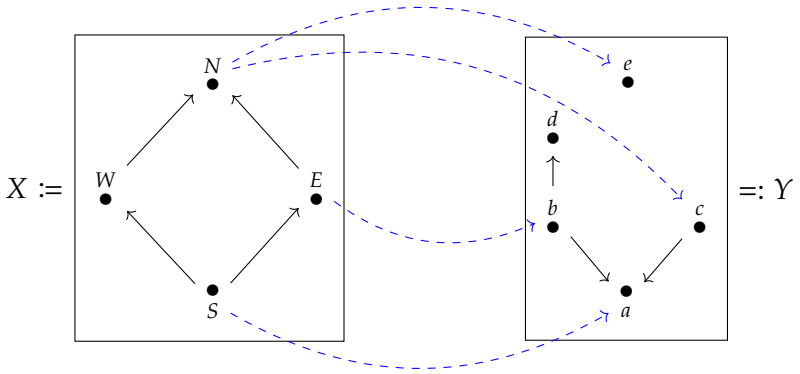
Exercise 4.6. Is it true that a **Bool**-profunctor, as in Definition 4.4 is exactly the same as a feasibility relation, as in Definition 4.1, once you peel back all the jargon? Or is there some subtle difference? ◊

We know that Definition 4.4 is quite abstract. But have no fear, we will take you through it in pictures.

Example 4.7 (**Bool**-profunctors and their interpretation as bridges). Let's discuss Definition 4.4 in the case $\mathcal{V} =$ **Bool**. One way to imagine a **Bool**-profunctor $\Phi\colon X \nrightarrow Y$ is in terms of building bridges between two cities. Recall that a poset (a **Bool**-category) can be drawn using a Hasse diagram. We'll think of the poset as a city, and each vertex in it as some point of interest. An arrow $A \to B$ in the Hasse diagram means that there exists a way to get from point A to point B in the city. So what's a profunctor?

A profunctor is just a bunch of bridges connecting points in one city to points in

another. Let's see a specific example. Here is a picture of a **Bool**-profunctor $\Phi\colon X \nrightarrow Y$:



Both X and Y are posets, e.g. with $W \leq N$ and $b \leq a$. With bridges coming from the profunctor in blue, one can now use both paths within the cities and the bridges get from points in city X to points in city Y. For example, since there is a path from N to e and E to a, we have $\Phi(N, e) = \texttt{true}$ and $\Phi(E, a) = \texttt{true}$. On the other hand, since there is no path from W to d, we have $\Phi(W, d) = \texttt{false}$.

In fact, one could put a box around this entire picture and see a new poset with $W \leq N \leq c \leq a$, etc. This is called the *collage* of Φ; we'll explore this in more detail in Section 4.3.3. ♦

Exercise 4.8. We can express Φ as a matrix where the (m, n)th entry is the value of $\Phi(m, n) \in \mathbb{B}$. Fill out the **Bool**-matrix:

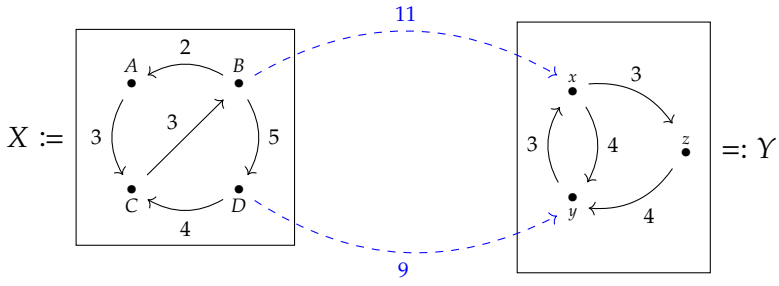
Φ	a	b	c	d	e
N	?	?	?	?	`true`
E	`true`	?	?	?	?
W	?	?	?	`false`	?
S	?	?	?	?	?

◊

We'll call this the *feasibility matrix* of Φ.

Example 4.9 (**Cost**-profunctors and their interpretation as bridges). Let's now consider **Cost**-profunctors. Again we can view these as bridges, but this time our bridges are labelled by their length. Recall from **????** that **Cost**-categories are Lawvere metric spaces, and can be depicted using weighted graphs. We'll think of such a weighted graph as a chart of distances between points in a city, and generate a **Cost**-profunctor by building a few bridges between the cities.

Here is a depiction of a **Cost**-profunctor $\Phi\colon X \nrightarrow Y$:



(4.5)

The distance from a point x in city X to a point y in city Y is given by the shortest path that runs from x through X, then across one of the bridges, and then through Y to the destination y. So for example

$$\Phi(B, x) = 11, \quad \Phi(A, z) = 20, \quad \Phi(C, y) = 17. \qquad \blacklozenge$$

Exercise 4.10. Fill out the **Cost**-matrix:

Φ	x	y	z
A	?	?	20
B	11	?	?
C	?	17	?
D	?	?	?

$\lozenge$

Remark 4.11 (Computing profunctors via matrix multiplication). We can give an algorithm for computing the above distance matrix using matrix multiplication. First, just like in **??**, we can begin with the labelled graphs in Eq. (4.5) and read off the matrices of arrow labels for X, Y, and Φ:

M_X	A	B	C	D
A	0	∞	∞	∞
B	2	0	∞	5
C	∞	3	0	∞
D	∞	∞	4	0

M_Φ	x	y	z
A	∞	∞	∞
B	11	∞	∞
C	∞	∞	∞
D	∞	9	∞

M_Y	x	y	z
x	0	4	3
y	3	0	∞
z	∞	4	0

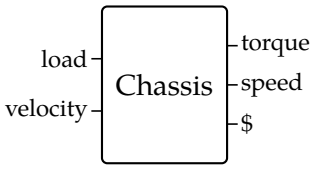
Recall from **??** that the matrix of distances d_Y for **Cost**-category X can be obtained by taking the matrix power of M_X with smallest entries, and similarly for Y. The matrix of distances for the profunctor Φ will be equal to $d_X * M_\Phi * d_Y$. In fact, since X has four elements and Y has three, we also know that $\Phi = M_X^4 * M_\Phi * M_Y^3$.

Exercise 4.12. Calculate $M_X^4 * M_\Phi * M_Y^3$, remembering to do matrix multiplication according to the $(\min, +)$-formula for matrix multiplication in the quantale **Cost**; see **??**.

Your answer should agree with what you got in Exercise 4.10; does it? $\lozenge$

4.2.3 Back to co-design diagrams

Each box in a co-design diagram has a left-hand and a right-hand side, which in turn consist of a collection of ports, which in turn are labeled by posets. For example, consider the chassis box below:

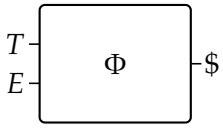


Its left side consists of two ports—one for load and one for velocity—and these are the functionality that the chassis produces. Its right side consists of three ports—one for torque, one for speed, and one for $—and these are the resources that the chassis requires. Each of these resources is to be taken as a poset. For example, load might be the poset $([0, \infty], \leq)$, where an element $x \in [0, \infty]$ represents the idea "I can handle any load up to x.", while $ might be the two-element poset {`up_to_$100`, `more_than_$100`}, where the first element of this set is less than the second.

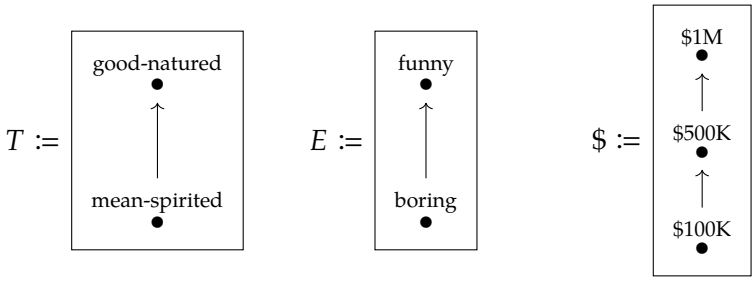
We then multiply—i.e. we take the product poset—of all posets on the left, and similarly for those on the right. The box then represents a feasibility relation between the results. For example, the chassis box above represents a feasibility relation

$$\text{Chassis}\colon \text{load} \times \text{velocity} \nrightarrow \text{torque} \times \text{speed} \times \$$$

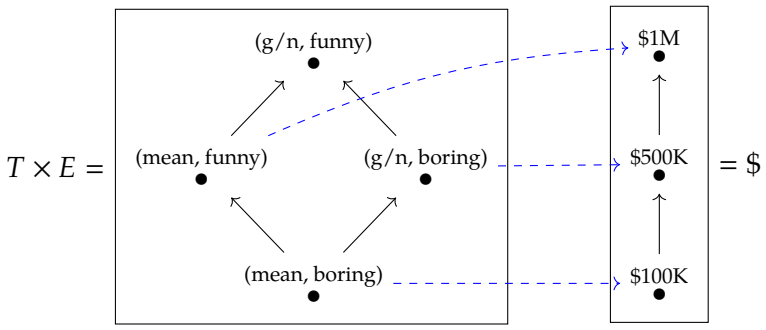
Let's walk through this a bit more concretely. Consider the design problem of filming a movie, where you must pit the tone and entertainment value against the cost. A feasibility relation describing this situation details what tone and entertainment value can be obtained at each cost; as such, it is described by a feasibility relation $\Phi\colon (T \times E) \nrightarrow \$$. We represent this by the box



where T, E, and $ are the posets drawn below:



A possible feasibility relation is then described by the profunctor



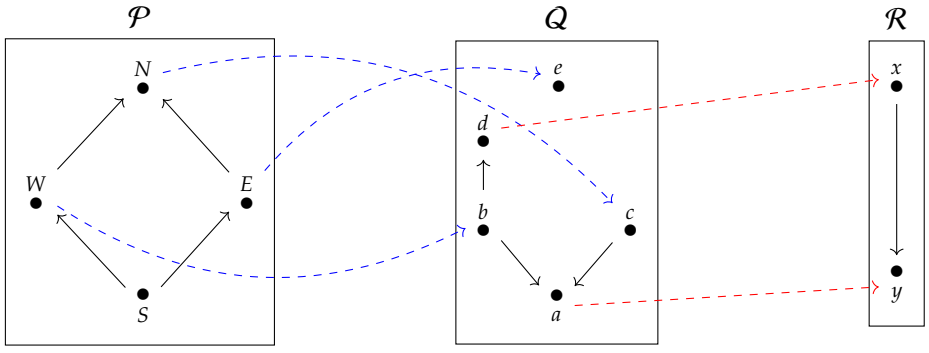
This says, for example, that a good-natured but boring movie costs $500K to produce (of course, the producers would also be happy to get $1M).

4.3 Categories of profunctors

There is a category **Feas** whose objects are posets and whose morphisms are feasibility relations. In order to describe it, we must give the composition formula and the identities, and prove that they satisfy the properties of being a category: unitality and associativity.

4.3.1 Composing profunctors

If feasibility relations are to be morphisms, we need to give a formula for composing two of them in series. Imagine you have cities $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ and you have bridges—and hence feasibility matrices—connecting these cities, say $\Phi\colon \mathcal{P} \nrightarrow \mathcal{Q}$ and $\Psi\colon \mathcal{Q} \nrightarrow \mathcal{R}$.



(4.6)

The feasibility matrices for Φ (in blue) and Ψ (in red) are:

Φ	a	b	c	d	e
N	true	false	true	false	false
E	true	false	true	false	true
W	true	true	true	true	false
S	true	true	true	true	true

Ψ	x	y
a	false	true
b	true	true
c	false	true
d	true	true
e	false	false

As in **??**, we personify a quantale as a navigator. So imagine a navigator is trying to give a feasibility matrix $\Phi.\Psi$ for getting from $\mathcal{P}$ to $\mathcal{R}$. How should this be done? Basically, for every pair $p \in \mathcal{P}$ and $r \in \mathcal{R}$, the navigator searches through $\mathcal{Q}$ for a way-point q, somewhere both to which we can get from p AND from which we can get to r. It is true that we can navigate from p to r if any q can be a way-point; this is a big OR over all possible q. The composition formula is thus:

$$(\Phi.\Psi)(p,q) := \bigvee_{q \in \mathcal{Q}} \Phi(p,q) \wedge \Psi(q,r). \tag{4.7}$$

But as we have said, this can be thought of as matrix multiplication. In our example, the result is

$\Phi.\Psi$	x	y
N	`false`	`true`
E	`false`	`true`
W	`true`	`true`
S	`true`	`true`

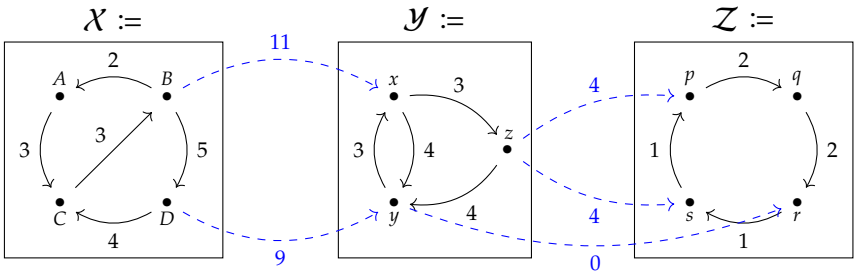
and one can check that this answers the question, "can you get from here to there" in Eq. (4.6): you can't get from N to x but you can get from N to y.

The formula (4.7) is written in terms of the quantale **Bool**, but it works for arbitrary commutative quantales. We give the following definition.

Definition 4.13. Let $\mathcal{V}$ be a commutative quantale, let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be $\mathcal{V}$-categories, and let $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$ and $\Psi\colon \mathcal{Y} \nrightarrow \mathcal{Z}$ be $\mathcal{V}$-profunctors. We define their *composite*, denoted $\Phi.\Psi\colon \mathcal{X} \nrightarrow \mathcal{Z}$ as the map given by the formula

$$(\Phi.\Psi)(p,r) = \bigvee_{q \in Q} \big(\Phi(p,q) \otimes \Psi(q,r)\big).$$

Exercise 4.14. Consider the **Cost**-profunctors $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$ and $\Psi\colon \mathcal{Y} \nrightarrow \mathcal{Z}$ shown below:



Fill in the matrix:

$\Phi.\Psi$	p	q	r	s
A	?	24	?	?
B	?	?	?	?
C	?	?	?	?
D	?	?	9	?

◊

4.3.2 The categories $\mathcal{V}$-Prof and Feas

A composition rule suggests a category, and there is indeed a category where the objects are **Bool**-categories and the morphisms are **Bool**-profunctors. To make this work more generally, however, we need to add one technical condition.

Recall that a poset is a skeletal poset if whenever $x \leq y$ and $y \leq x$, we have $x = y$. A skeletal poset is also known as a partially ordered set. We say a quantale is skeletal if its underlying poset is skeletal; **Bool** and **Cost** are skeletal quantales.

Theorem 4.15. *For any skeletal commutative quantale $\mathcal{V}$,[1] there is a category $\mathbf{Prof}_{\mathcal{V}}$ whose objects are $\mathcal{V}$-categories $\mathcal{X}$, whose morphisms are $\mathcal{V}$-profunctors $\mathcal{X} \nrightarrow \mathcal{Y}$, and with composition defined as in Definition 4.13.*

Definition 4.16. We define $\mathbf{Feas} := \mathbf{Prof}_{\mathbf{Bool}}$.

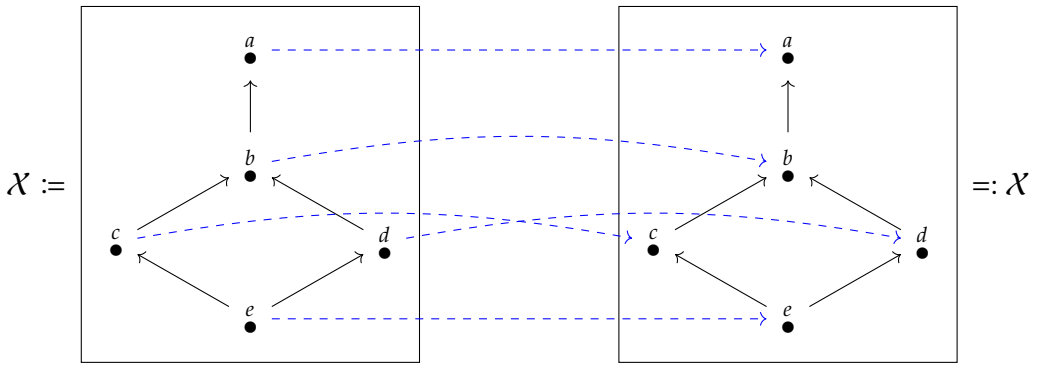
At this point perhaps you have two questions in mind. What are the identity morphisms? And why did we need to specialize to skeletal quantales? It turns out these two questions are closely related.

Define the *unit profunctor* $\mathrm{U}_{\mathcal{X}}$ on a $\mathcal{V}$-category $\mathcal{X}$ by the formula

$$\mathrm{U}_{\mathcal{X}}(x, y) := \mathcal{X}(x, y). \tag{4.8}$$

How do we interpret this? Recall that, by **??**, $\mathcal{X}$ already assigns to each pair of elements $x, y \in \mathcal{X}$ an hom-object $\mathcal{X}(x, y) \in \mathcal{V}$. The unit profunctor $\mathrm{U}_{\mathcal{X}}$ just assigns each pair (x, y) that same object.

In the **Bool** case the unit profunctor on some poset $\mathcal{X}$ can be drawn like this:



Obviously, composing a feasibility relation with with the unit leaves it unchanged, which is the content of Lemma 4.18.

Exercise 4.17. Choose a not-too-simple **Cost**-category $\mathcal{X}$. Give a bridge-style diagram for the unit profunctor $U_{\mathcal{X}} \colon \mathcal{X} \nrightarrow \mathcal{X}$. ◊

Lemma 4.18. *Composing any profunctor $\Phi \colon \mathcal{P} \to \mathcal{Q}$ with either unit profunctor, $\mathrm{U}_{\mathcal{P}}$ or $\mathrm{U}_{\mathcal{Q}}$, returns Φ:*

$$\mathrm{U}_{\mathcal{P}}.\Phi = \Phi = \Phi.\mathrm{U}_{\mathcal{Q}}$$

[1] From here on, as in **??**, whenever we speak of quantale we mean commutative quantales.

Proof. We show that $\mathrm{U}_{\mathcal{P}}.\Phi = \Phi$ holds; proving $\Phi = \Phi.\mathrm{U}_{\mathcal{Q}}$ is similar. Fix $p \in P$ and $q \in Q$. Since $\mathcal{V}$ is skeletal, to prove the equality it's enough to show $\Phi \leq \mathrm{U}_{\mathcal{P}}.\Phi$ and $\mathrm{U}_{\mathcal{P}}.\Phi \leq \Phi$. We have one direction:

$$\Phi(p,q) = I \otimes \Phi(p,q) \leq \mathcal{P}(p,p) \otimes \Phi(p,q) \leq \bigvee_{p_1 \in P} \big(\mathcal{P}(p,p_1) \otimes \Phi(p_1,q)\big) = (\mathrm{U}_{\mathcal{P}}.\Phi)(p,q). \tag{4.9}$$

For the other direction, we must show $\bigvee_{p_1 \in P} \big(\mathcal{P}(p,p_1) \otimes \Phi(p_1,q)\big) \leq \Phi(p,q)$. But by definition of join, this holds iff $\mathcal{P}(p,p_1) \otimes \Phi(p_1,q) \leq \Phi(p,q)$ is true for each $p_1 \in \mathcal{P}$. This follows from **??** and Definition 4.4:

$$\mathcal{P}(p,p_1) \otimes \Phi(p_1,q) = \mathcal{P}(p,p_1) \otimes \Phi(p_1,q) \otimes I \leq \mathcal{P}(p,p_1) \otimes \Phi(p_1,q) \otimes \mathcal{Q}(q,q) \leq \Phi(p,q). \quad \square$$

Exercise 4.19. 1. Justify each of the four steps $(=, \leq, \leq, =)$ in Eq. (4.9).

2. In the case $\mathcal{V} = \mathbf{Bool}$, we can directly show each of the four steps in Eq. (4.9) is actually an equality. How? ◊

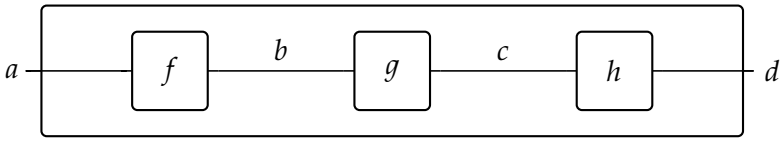
Composition of profunctors is also associative; we leave the proof to you.

Lemma 4.20. *Serial composition of profunctors is associative: given profunctors* $\Phi\colon \mathcal{P} \to \mathcal{Q}$, $\Psi\colon \mathcal{Q} \to \mathcal{R}$, *and* $\Upsilon\colon \mathcal{R} \to \mathcal{S}$, *we have*

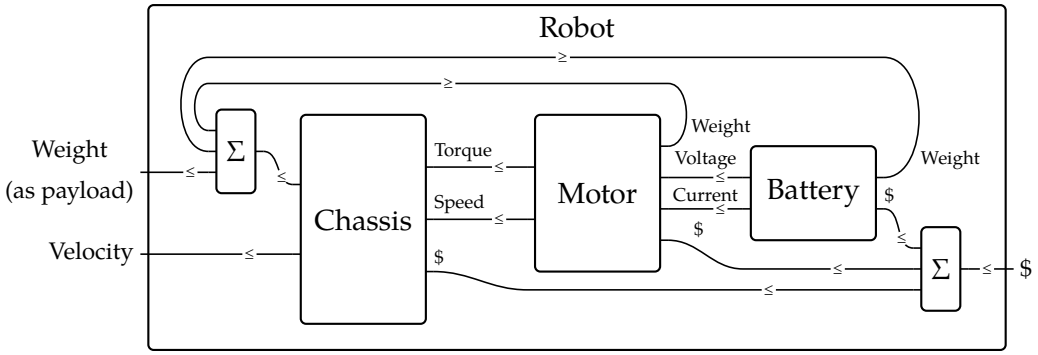
$$(\Phi.\Psi).\Upsilon = \Phi.(\Psi.\Upsilon).$$

Exercise 4.21. Prove Lemma 4.20. (Hint: remember to use the fact that $\mathcal{V}$ is skeletal.) ◊

So, feasibility relations form a category. Since this is the case, we can describe feasibility relations using string diagrams for categories. Recall, however, string diagrams for categories are very simple. Indeed, each box can only have one input and one output, and they're connected in a line:



On the other hand, we have seen that feasibility relations are the building blocks of co-design problems, and we know that co-design problems can be depicted in a much richer way, for example:



This hints that the category **Feas** has more structure. We saw wiring diagrams where boxes can have multiple inputs and outputs in **??**; there they depicted morphisms in a monoidal poset. On other hand the boxes in the wiring diagrams of **??** could not have labels, like the boxes in a co-design problem, and similarly, we know that **Feas** is a proper category, not just a poset. To understand these diagrams then, we will have to introduce a new structure. This will be a *categorified* monoidal poset; these are known, not surprisingly, as *monoidal categories*.

Remark 4.22. While we have chosen to define $\mathbf{Prof}_{\mathcal{V}}$ only for skeletal (commutative) quantales in Theorem 4.15, it is not too hard to work with non-skeletal ones. There are two straightforward ways to do this. First, we might let the morphisms of $\mathbf{Prof}_{\mathcal{V}}$ be isomorphism classes of $\mathcal{V}$-profunctors. This is analogous to the trick we will use when defining the category $\mathbf{Cospan}_C$ in **??**. Second, we might relax what we mean by category, only requiring composition to be unital and associative 'up to isomorphism'. This is also a type of categorification.

In the next section we'll discuss categorification and introduce monoidal categories. First though, we finish this section by discussing why profunctors are called profunctors, and by formally introducing the notation for profunctors called a *collage*.

4.3.3 Fun profunctor facts: companions, conjoints, collages

Companions and conjoints

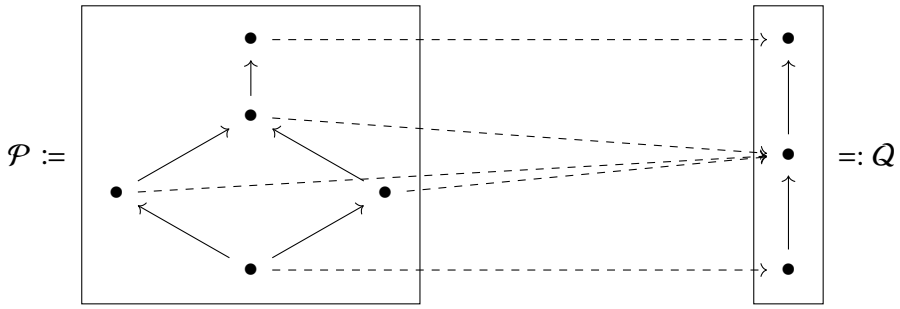
Recall that a poset is a **Bool**-category and a monotone map is a **Bool**-functor. We said above that a profunctor is a generalization of a functor; how so?

In fact, every $\mathcal{V}$-functor gives rise to two $\mathcal{V}$-profunctors, called the companion and the conjoint.

Definition 4.23. Let $F\colon \mathcal{P} \to \mathcal{Q}$ be a $\mathcal{V}$-functor. The *companion of* F, denoted $\widehat{F}\colon \mathcal{P} \nrightarrow \mathcal{Q}$ and the *conjoint of* F, denoted $\check{F}\colon \mathcal{Q} \nrightarrow \mathcal{P}$ are defined to be the following $\mathcal{V}$-profunctors:

$$\widehat{F}(p,q) \coloneqq \mathcal{Q}(F(p),q) \quad \text{and} \quad \check{F}(q,p) \coloneqq \mathcal{Q}(q,F(p))$$

Let's consider the **Bool** case again. One can think of a monotone map $F\colon \mathcal{P} \to \mathcal{Q}$ as a bunch of arrows, one coming out of each vertex $p \in P$ and landing at some vertex $F(p) \in Q$.



This looks like the pictures of bridges connecting cities, and if one regards the above picture in that light, they are seeing the companion $\widehat{F}$. But now mentally reverse every dotted arrow, and the result would be bridges $\mathcal{Q}$ to $\mathcal{P}$. This is a profunctor $\mathcal{Q} \nrightarrow \mathcal{P}$! We call it $\check{F}$.

Example 4.24. For any poset $\mathcal{P}$, there is an identity functor $\mathrm{id}\colon \mathcal{P} \to \mathcal{P}$. Its companion and conjoint agree $\widehat{\mathrm{id}} = \check{\mathrm{id}}\colon \mathcal{P} \nrightarrow \mathcal{P}$. The resulting profunctor is in fact the unit profunctor, $\mathrm{U}_{\mathcal{P}}$ as defined in Eq. (4.8). ♦

Exercise 4.25. Explain why the companion $\widehat{\mathrm{id}}$ of $\mathrm{id}\colon \mathcal{P} \to \mathcal{P}$ really has the formula given in Eq. (4.8). ◊

Example 4.26. Consider the function $+\colon \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, sending a triple (a, b, c) of real numbers to $a + b + c \in \mathbb{R}$. This function is monotonic, because if $(a, b, c) \leq (a', b', c')$—i.e. if $a \leq a'$ and $b \leq b'$, and $c \leq c'$—then obviously $a + b + c \leq a' + b' + c'$. Thus it has a companion and a conjoint.

Its companion $\widehat{+}\colon (\mathbb{R} \times \mathbb{R} \times \mathbb{R}) \nrightarrow \mathbb{R}$ is the function that sends (a, b, c, d) to `true` if $a + b + c \leq d$ and to `false` otherwise. ♦

Exercise 4.27. Let $+\colon \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be as in Example 4.26. What is its conjoint $\check{+}$? ◊

Remark 4.28 ($\mathcal{V}$-Adjoints). Recall from **??** the definition of Galois connection between posets $\mathcal{P}$ and $\mathcal{Q}$. The definition of adjoint can be extended from the **Bool**-enriched setting (of posets and monotone maps) to the $\mathcal{V}$-enriched setting for arbitrary monoidal posets $\mathcal{V}$. In that case, the definition of a $\mathcal{V}$-adjunction is a pair of $\mathcal{V}$-functors $F\colon \mathcal{P} \to \mathcal{Q}$ and $G\colon \mathcal{Q} \to \mathcal{P}$ such that the following holds for all $p \in P$ and $q \in Q$.

$$\mathcal{P}(p, G(q)) \cong \mathcal{Q}(F(p), q) \tag{4.10}$$

Exercise 4.29. Let $\mathcal{V}$ be a skeletal quantale, let $\mathcal{P}$ and $\mathcal{Q}$ be $\mathcal{V}$-categories, and let $F\colon \mathcal{P} \to \mathcal{Q}$ and $G\colon \mathcal{Q} \to \mathcal{P}$ be $\mathcal{V}$-functors.

1. Show that F and G are $\mathcal{V}$-adjoints (as in Eq. (4.10)) if and only if the companion of the former equals the conjoint of the latter: $\widehat{F} = \check{G}$.
2. Use this to prove that $\widehat{\mathrm{id}} = \check{\mathrm{id}}$, as was stated in Example 4.24. ◊

Collage of a profunctor

We have been drawing profunctors as bridges connecting cities. One may get an inkling that given a $\mathcal{V}$-profunctor $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$ between $\mathcal{V}$-categories $\mathcal{X}$ and $\mathcal{Y}$, we have turned Φ into a some sort of new $\mathcal{V}$-category that has $\mathcal{X}$ on the left and $\mathcal{Y}$ on the right. This works for any $\mathcal{V}$ and profunctor Φ, and is called the collage construction.

Definition 4.30. Let $\mathcal{V}$ be a quantale, let $\mathcal{X}$ and $\mathcal{Y}$ be $\mathcal{V}$-categories, and let $\Phi\colon \mathcal{X} \nrightarrow \mathcal{Y}$ be a $\mathcal{V}$-profunctor. The *collage of* Φ, denoted $\mathbf{Col}(\Phi)$ is the $\mathcal{V}$-category defined as follows:

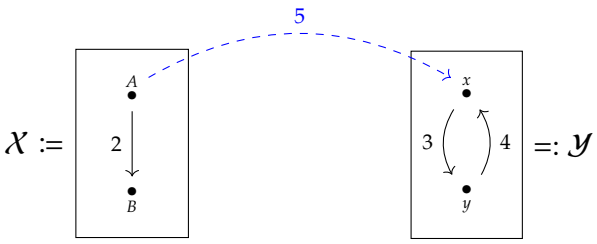
1. $\mathrm{Ob}(\mathbf{Col}(\Phi)) := \mathrm{Ob}(\mathcal{X}) \sqcup \mathrm{Ob}(\mathcal{Y})$;

2. For any $a, b \in \mathrm{Ob}(\mathbf{Col}(\Phi)$, define $\mathbf{Col}(\Phi)(a,b) \in \mathcal{V}$ to be

$$\mathbf{Col}(\Phi)(a,b) := \begin{cases} \mathcal{X}(a,b) & \text{if } a,b \in \mathcal{X} \\ \Phi(a,b) & \text{if } a \in \mathcal{X}, b \in \mathcal{Y} \\ \emptyset & \text{if } a \in \mathcal{Y}, b \in \mathcal{X} \\ \mathcal{Y}(a,b) & \text{if } a,b \in \mathcal{Y} \end{cases}$$

There are obvious functors $i_{\mathcal{X}} \colon \mathcal{X} \to \mathbf{Col}(\Phi)$ and $i_{\mathcal{Y}} \colon \mathcal{Y} \to \mathbf{Col}(\Phi)$, sending each object and morphism to "itself", called *collage inclusions*.

Example 4.31. For example, consider the following picture of a **Cost**-profunctor $\Phi \colon \mathcal{X} \nrightarrow \mathcal{Y}$:



It corresponds to the following matrices

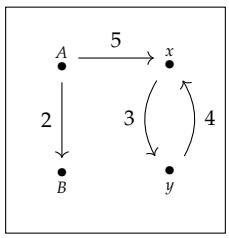
$\mathcal{X}$	A	B
A	0	2
B	∞	0

Φ	x	y
A	5	8
B	∞	∞

$\mathcal{Y}$	x	y
x	0	3
y	4	0

A generalized Hasse diagram of the collage can be obtained by simply taking the union of the Hasse diagrams for $\mathcal{X}$ and $\mathcal{Y}$, and adding in the bridges as arrows. Given the above profunctor Φ, we draw the Hasse diagram for $\mathbf{Col}(\Phi)$ below left, and the **Cost**-matrix representation of the resulting **Cost**-category on the right:

$\mathbf{Col}(\Phi) =$



$\mathbf{Col}(\Phi)$	A	B	x	y
A	0	2	5	8
B	∞	0	∞	∞
x	0	0	0	3
y	0	0	4	0

♦

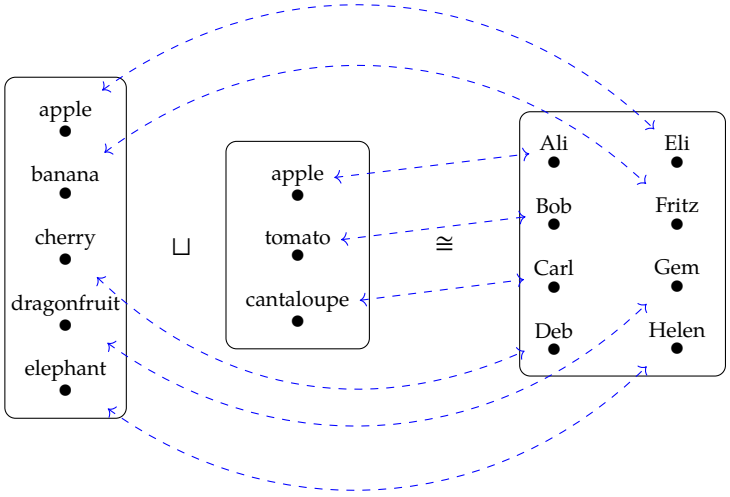
4.4 Categorification

Here we switch gears, to discuss a general concept called *categorification*. We will begin again with the basics, categorifying several of the notions we've encountered already. The goal is to define compact closed categories and their feedback-style wiring diagrams. At that point we will return to the story of co-design, and $\mathcal{V}$-profunctors in general, and show that they do in fact form a compact closed category, and thus interpret the diagrams we've been drawing since Eq. (4.1).

4.4.1 The basic idea of categorification

The general idea of categorification is that we take a thing we know and add structure to it, so that what were formerly *properties* become *structures*. We do this in such a way that we can recover the thing we categorified by forgetting this new structure. This is rather vague; let's give an example.

Basic arithmetic concerns properties of the natural numbers $\mathbb{N}$, such as the fact that $5 + 3 = 8$. One way to categorify $\mathbb{N}$ is to use the category **FinSet** of finite sets and functions. To obtain a categorification, we replace the brute 5, 3, and 8 with sets of that many elements, say $\overline{5} = \{\text{apple}, \text{banana}, \text{cherry}, \text{dragonfruit}, \text{elephant}\}$, $\overline{3} = \{\text{apple}, \text{tomato}, \text{cantaloupe}\}$, and $\overline{8} = \{\text{Ali}, \text{Bob}, \text{Carl}, \text{Deb}, \text{Eli}, \text{Fritz}, \text{Gem}, \text{Helen}\}$ respectively. We also replace $+$ with disjoint union of sets $\sqcup$, and the brute property of equality with the structure of an isomorphism. What makes this a good categorification is that, having made these replacements, the analogue of $5+3 = 8$ is still true: $\overline{5} \sqcup \overline{3} \cong \overline{8}$.



In this categorified world, we have more structure available to talk about the relationships between objects, so we can be more precise about how they relate to each other. Thus it's not the case that $\overline{5} \sqcup \overline{3}$ is *equal* to our chosen eight-element set $\overline{8}$, but more precisely that there exists an invertible function comparing the two, showing that they are isomorphic in the *category* **FinSet**.

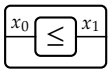
Note that in the above construction we made a number of choices; here we must beware. Choosing a good categorification, like designing a good algebraic structure like that of posets or quantales, is part of the art of mathematics. There is no prescribed way to categorify, and the success of a chosen categorification is often empirical: its richer structure allows us more insights into the subject we want to model.

As another example, an empirically pleasing way to categorify posets is to categorify them as, well, categories. In this case, rather than the brute property "there exists a morphism $a \to b$", denoted $a \leq b$ or $\mathcal{P}(a, b) = \texttt{true}$, we instead say "here is a set of

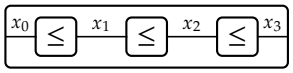
morphisms $a \to b$". We get a hom-set rather than a hom-Boolean. In fact—to state this in a way straight out of the primordial ooze—just as posets are **Bool**-categories, ordinary categories are actually **Set**-categories.

4.4.2 A reflection on wiring diagrams

Suppose we have a poset. We introduced a very simple sort of wiring diagram in **??**. These allowed us to draw a box

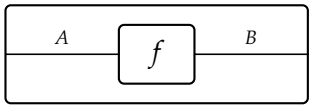


whenever $x_0 \leq x_1$. Chaining these together, we could prove facts in our poset. For example

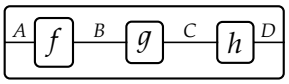


provides a proof that $x_0 \leq x_3$ (the exterior box) using three facts (the interior boxes), $x_0 \leq x_1$, $x_1 \leq x_2$, and $x_2 \leq x_3$.

As categorified posets, categories have basically the same sort of wiring diagram as posets—namely sequences of boxes inside a box. But since we have replace the fact that $x_0 \leq x_1$ with the structure of a *set* of morphisms, we need to be able to label our boxes with morphism names:



Suppose given additional morphisms $g\colon A \to B$, and $h\colon C \to D$. Representing these each as boxes like we did for f, we might be tempted to stick them together to form a new box:

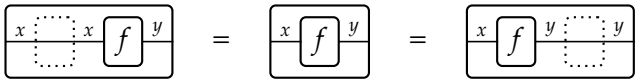


Ideally this would also be a morphism in our category: after all, we have said that we can represent morphisms with boxes with one input and one output. But wait, you say! We don't know which morphism it is. Is it $f.(g.h)$? Or $(f.g).h$? It's good that you are so careful. Luckily, we are saved by the properties that a category must have. Associativity says $f.(g.h) = (f.g).h$, so it doesn't matter which way we chose to try to decode the box.

Similarly, the identity morphism on an object x is drawn as on the left below, but we will see that it is not harmful to draw id_x in any of the following three ways:

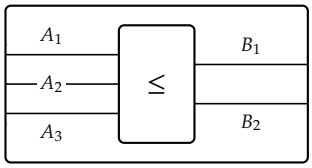
By **??** the morphisms in a category satisfy two properties, called the unitality property and the associativity property. The unitality says that $\mathrm{id}_x . f = f = f . \mathrm{id}_y$ for any

$f : x \to y$. In terms of diagrams this would say

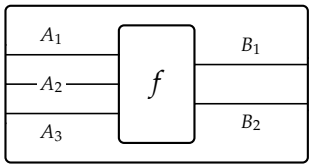


This means you can insert or discard any identity morphism you see in a wiring diagram. From this perspective, the coherence laws of a category—that is, the associativity law and the unitality law—are precisely what are needed to ensure we can draw these diagrams without ambiguity.

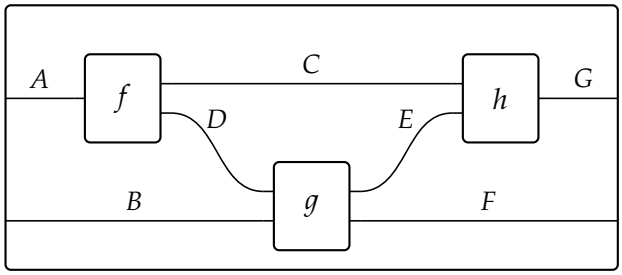
In **??**, we also saw wiring diagrams for monoidal posets. Here we were allowed to draw boxes which can have multiple typed inputs and outputs, but with no label:



If we combine these ideas, we will obtain a categorification of symmetric monoidal posets: symmetric monoidal categories. A symmetric monoidal category is an algebraic structure in which we have labelled boxes with have multiple typed inputs and outputs:



Furthermore, a symmetric monoidal category has a composition rule and a monoidal product, which permit us to combine these boxes to interpret diagrams like this:



Finally, this structure must obey coherence laws, analogous to associativity and unitality in categories, that allow such diagrams to be unambiguously interpreted. In the next section we will be a bit more formal, but it is useful to keep in mind that, when we say our data must be well behaved, this is all we mean.

4.4.3 Monoidal categories

We defined $\mathcal{V}$-categories, for a symmetric monoidal poset $\mathcal{V}$ in **??**. Just like posets turned out to be special kinds of categories (see **??**), monoidal posets are special kinds

of monoidal categories. And just like we can consider $\mathcal{V}$-categories for a monoidal poset, we can also consider $\mathcal{V}$-categories when $\mathcal{V}$ is a monoidal category. This is another sort of categorification.

We will soon meet the monoidal category (**Set**, $\{1\}, \times$). The monoidal product will take two sets, S and T, and return the set $S \times T = \{(s,t) \mid s \in S, t \in T\}$. But whereas for monoidal posets we had the brute associative property $(p \otimes q) \otimes r = p \otimes (q \otimes r)$, the corresponding idea in **Set** is not quite true:

$$\begin{aligned} & S \times (T \times U) := \big\{ \big(s,(t,u)\big) \,\big|\, s \in S, t \in T, u \in U \big\} \\ =^{?} \quad & (S \times T) \times U := \big\{ \big((s,t),u\big) \,\big|\, s \in S, t \in T, u \in U \big\}. \end{aligned}$$

They are slightly different sets: the first contains pairs consisting of an elements in S and an element in $T \times U$, while the second contains pairs consisting of an element in $S \times T$ and an element in U. The sets are not equal, but they are clearly isomorphic, i.e. the difference between them is "just a matter of bookkeeping". We thus need a structure—a bookkeeping isomorphism—to keep track of the associativity:

$$\alpha_{s,t,u} \colon \{(s,(t,u)) \mid s \in S, t \in T, u \in U\} \xrightarrow{\cong} \{((s,t),u)) \mid s \in S, t \in T, u \in U\}.$$

There are a couple things to mention before we dive into these ideas. First, just because you replace brute things and properties with structures, it does not mean that you no longer have brute things and properties: new ones emerge! Not only that, but second, the new brute stuff tends to be more complex than what you started with. For example, above we replaced the associativity equation with an isomorphism $\alpha_{s,t,u}$, but now we need a more complex property to ensure that α behaves reasonably! The only way out of this morass is to add infinitely much structure, which leads one to "∞-categories", but we will not discuss that here.

Instead, we will continue with our categorification of monoidal posets, starting with a rough definition of symmetric monoidal categories. It's rough in the sense that we suppress the technical bookkeeping, hiding it under the name "well behaved".

Rough Definition 4.32. Let C be a category. A *symmetric monoidal structure* on C consists of the following constituents:

(i) an object $I \in \mathrm{Ob}(C)$ called the *monoidal unit*, and
(ii) a functor $\otimes \colon C \times C \to C$, called the *monoidal product*

subject to well-behaved, natural isomorphisms

(a) $\lambda_c \colon I \otimes c \cong c$ for every $c \in \mathrm{Ob}(C)$,
(b) $\rho_c \colon c \otimes I \cong c$ for every $c \in \mathrm{Ob}(C)$,
(c) $\alpha_{c,d,e} \colon (c \otimes d) \otimes e \cong c \otimes (d \otimes e)$ for every $c, d, e \in \mathrm{Ob}(C)$, and
(d) $\sigma_{c,d} \colon c \otimes d \cong d \otimes c$ for every $c, d \in \mathrm{Ob}(C)$, such that $\sigma \circ \sigma = \mathrm{id}$.

A category equipped with a symmetric monoidal structure is called a *symmetric monoidal category*.

Remark 4.33. If the isomorphisms in (a), (b), and (c)—but *not* (d)—are replaced by equalities, then we say that the monoidal structure is *strict*, and this is a complete (non-rough) definition. In fact, symmetric strict monoidal categories are almost the same thing as symmetric monoidal categories. Ask a friendly expert category theorist to explain to you how!

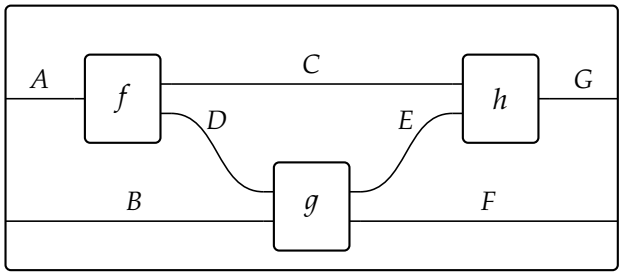
Exercise 4.34. Check that monoidal categories generalize monoidal posets: a monoidal poset is a monoidal category $(\mathcal{P}, I, \otimes)$ where, for every $p, q \in \mathcal{P}$, the set $\mathcal{P}(p, q)$ has at most one element. ◊

Example 4.35. As we said above, there is a monoidal structure on **Set** where the monoidal unit is some choice of singleton set, say $I := \{1\}$, and the monoidal product is $\otimes := \times$. What it means that $\times$ is a functor is that:

- For any pair of objects, i.e. sets, $(S, T) \in \mathrm{Ob}(\mathbf{Set} \times \mathbf{Set})$, one obtains a set $(S \times T) \in \mathrm{Ob}(\mathbf{Set})$. We know what it is: the set of pairs $\{(s, t) \mid s \in S, t \in T\}$.
- For any pair of morphisms, i.e. functions, $f : S \to S'$ and $g : T \to T'$, one obtains a function $(f \times g) : (S \times T) \to (S' \times T')$. It works pointwise: $(f \times g)(s, t) := (f(s), g(t))$.
- These should preserve identities: $\mathrm{id}_S \times \mathrm{id}_T = \mathrm{id}_{S \times T}$ for any sets S, T.
- These should preserve composition: for any functions $S \xrightarrow{f} S' \xrightarrow{f'} S''$ and $T \xrightarrow{g} T' \xrightarrow{g'} T''$, one has
$$(f \times g).(f' \times g') = (f.g) \times (f'.g').$$

The four conditions, (a), (b), (c), and (d) give isomorphisms $\{1\} \times S \cong S$, etc. These maps are obvious in the case of **Set**, e.g. the function $\{(1, s) \mid s \in S\} \to S$ sending $(1, s)$ to s. We have been calling such things bookkeeping. ♦

Exercise 4.36. Consider the monoidal category $(\mathbf{Set}, 1, \times)$, together with the diagram



Suppose that $A = B = C = D = F = G = \mathbb{Z}$ and $E = \mathbb{B} = \{\texttt{true}, \texttt{false}\}$, and suppose that $f_C(a) = |a|$, $f_D(a) = a * 5$, $g_E(d, b) = d \leq b$, $g_F(d, b) = d - b$, and $h(c, e) =$ if e then c else $1 - c$.

1. What are $g_E(5, 3)$ and $g_F(5, 3)$?
2. What are $g_E(3, 5)$ and $g_F(3, 5)$?
3. What is $h(5, \texttt{true})$?
4. What is $h(-5, \texttt{true})$?
5. What is $h(-5, \texttt{false})$?

The whole diagram now defines a function $A \times B \to G \times F$; call it q.

6. What are $q_G(-2,3)$ and $q_F(-2,3)$?
7. What are $q_G(2,3)$ and $q_F(2,3)$? ◊

We will see more monoidal categories throughout the remainder of this book.

4.4.4 Categories enriched in a symmetric monoidal category

We will not need this again, but had promised to explain why $\mathcal{V}$-categories, where $\mathcal{V}$ is a symmetric monoidal poset, deserve to be seen as types of categories. The reason, as we have hinted, is that categories should really be called **Set**-categories. But wait, **Set** is not a poset! We'll have to generalize—categorify—$\mathcal{V}$-categories.

We now give a rough definition of categories enriched in a symmetric monoidal category $\mathcal{V}$. As in Definition 4.32, we suppress some technical parts in this sketch, hiding them under the name "usual associative and unital laws".

Rough Definition 4.37. Let $\mathcal{V}$ be a symmetric monoidal category, as in Definition 4.32. To specify a *category enriched in* $\mathcal{V}$, or a $\mathcal{V}$*-category*, denoted $\mathcal{X}$,

(i) one specifies a collection $\mathrm{Ob}(\mathcal{X})$, elements of which are called *objects*;
(ii) for every pair $x, y \in \mathrm{Ob}(\mathcal{X})$, one specifies an object $\mathcal{X}(x,y) \in \mathcal{V}$, called the *hom-object* for x, y;
(iii) for every $x \in \mathrm{Ob}(\mathcal{X})$, one specifies a morphism $\mathrm{id}_x : I \to \mathcal{X}(x,x)$ in $\mathcal{V}$, called the *identity element*;
(iv) for each $x, y, z \in \mathrm{Ob}(\mathcal{X})$, one specifies a morphism $. : \mathcal{X}(x,y) \otimes \mathcal{X}(y,z) \to \mathcal{X}(x,z)$, called the *composition morphism*.

These constituents are required to satisfy the usual associative and unital laws.

The precise, non-rough, definition can be found in other sources, e.g. [**Nlab:symmeric-monoidal-category**] [**wiki:Symmetric-monoidal-category**], [**Kelly:1982a**].

Exercise 4.38. Recall from Example 4.35 that $\mathcal{V} = (\mathbf{Set}, \{1\}, \times)$ is a symmetric monoidal category. This means we can apply Definition 4.37. Does the (rough) definition roughly agree with the definition of category given in **??**? Or is there a subtle difference? ◊

Remark 4.39. We first defined $\mathcal{V}$-categories in **??**, where $\mathcal{V}$ was required to be a monoidal poset. To check we're not abusing our terms, it's a good idea to make sure that $\mathcal{V}$-categories as per **??** are still $\mathcal{V}$-categories as per Definition 4.37.

The first thing to observe is that every symmetric monoidal poset is a symmetric monoidal category (Exercise 4.34). So given a symmetric monoidal poset $\mathcal{V}$, we can apply Definition 4.37. The required data (i) and (ii) then get us off to a good start: both definitions of $\mathcal{V}$-category require objects and hom-objects, and they are specified in the same way. On the other hand, Definition 4.37 requires two additional pieces of data: (iii) identity elements and (iv) composition morphisms. Where do these come from?

In the case of posets, there is at most one morphism between any two objects, so we do not need to choose an identity element and a composition morphism. Instead,

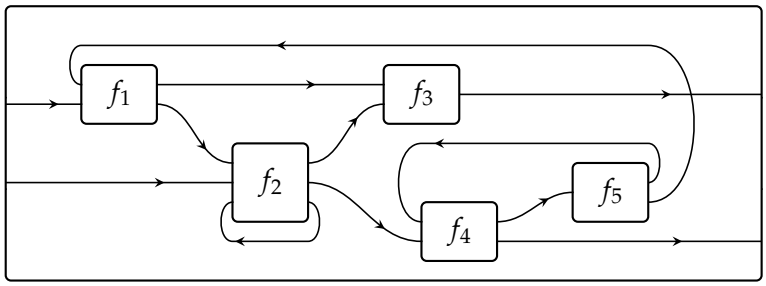
we just need to make sure that an identity element and a composition morphism exist. This is exactly what properties (a) and (b) of **??** say.

For example, the requirement (iii) that a $\mathcal{V}$-category $\mathcal{X}$ has a chosen identity element $\mathrm{id}_x \colon I \to \mathcal{X}(x,x)$ for the object x simply becomes the requirement (a) that $I \leq \mathcal{X}(x,x)$ is true in $\mathcal{V}$. This is typical of the story of categorification: what were mere properties in **??** become structures in Definition 4.37.

Exercise 4.40. What are identity elements in Lawvere metric spaces (that is, **Cost**-categories)? How do we interpret this in terms of distances? ◊

4.5 Profunctors form a compact closed category

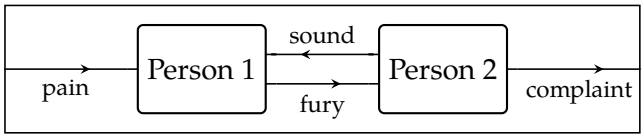
In this section we will define compact closed categories and show that **Feas**, and more generally $\mathcal{V}$-profunctors, form such a thing. Compact-closed categories are monoidal categories whose wiring diagrams allow feedback. The wiring diagrams look like this:



(4.11)

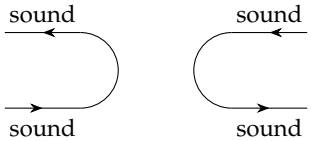
It's been a while since we thought about co-design, but these were the kinds of wiring diagrams we drew, e.g. connecting the chassis, the motor, and the battery in Eq. (4.1). Compact closed categories are symmetric monoidal categories, with a bit more structure that allow us to formally interpret the sorts of feedback that occur in co-design problems. This same structure shows up in many other fields, including quantum mechanics and dynamical systems.

In **????** we discussed various flavors of wiring diagrams, including those with icons for splitting and terminating wires. For compact-closed categories, our additional icons allow us to bend outputs into inputs, and vice versa. To keep track of this, however, we draw arrows on our wire, which can either point forwards or backwards. For example, we can draw this



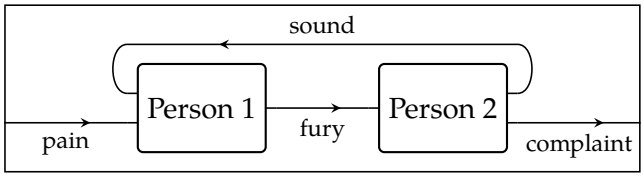
(4.12)

We then add icons—called a cap and a cup—allowing any wire to reverse direction from forwards to backwards and from backwards to forwards.



(4.13)

Thus we can draw the following



and its meaning is equivalent to that of Eq. (4.12).

We will begin by giving the axioms for a compact closed category. Then we will look again at feasibility relations in co-design—and more generally at enriched profunctors—and show that they indeed form a compact closed category.

4.5.1 Compact closed categories

As we said, compact closed categories are generalizations of symmetric monoidal categories (see Definition 4.32).

Definition 4.41. Let $(C, I, \otimes)$ be a symmetric monoidal category, and $c \in \mathrm{Ob}(C)$ an object. A *dual for* c consists of three consituents

(i) an object $c^* \in \mathrm{Ob}(C)$, called the *dual of* c,
(ii) a morphism $\eta_c \colon I \to c^* \otimes c$, called the *unit for* c,
(iii) a morphism $\epsilon_c \colon c \otimes c^* \to I$, called the *counit for* c.

These are required to satisfy two equations for every $c \in \mathrm{Ob}(C)$, which we draw as commutative diagrams:

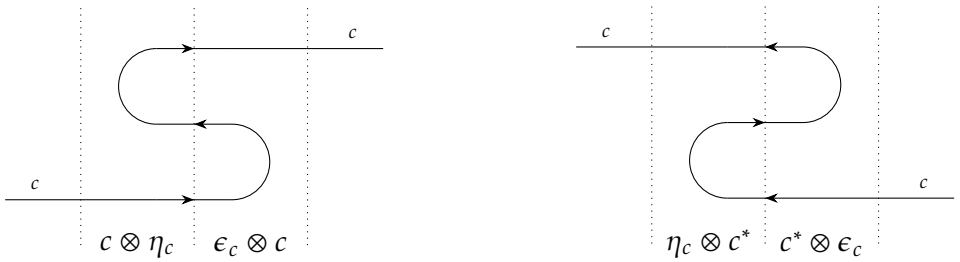
$$\begin{array}{ccc} c & = & c \\ \cong\downarrow & & \uparrow\cong \\ c \otimes I & & I \otimes c \\ {\scriptstyle c\otimes\eta_c}\downarrow & & \uparrow{\scriptstyle \epsilon_c\otimes c} \\ c \otimes (c^* \otimes c) & \xrightarrow[\cong]{} & (c \otimes c^*) \otimes c \end{array} \qquad \begin{array}{ccc} c^* & = & c^* \\ \cong\downarrow & & \uparrow\cong \\ I \otimes c^* & & c^* \otimes I \\ {\scriptstyle \eta_c\otimes c^*}\downarrow & & \uparrow{\scriptstyle c^*\otimes\epsilon_c} \\ (c^* \otimes c) \otimes c^* & \xrightarrow[\cong]{} & c^* \otimes (c \otimes c^*) \end{array} \tag{4.14}$$

These equations are sometimes called the *snake equations*.

If for every object $c \in \mathrm{Ob}(C)$ there exists a dual c^* for c, then we say that $(C, I, \otimes)$ is *compact closed*.

In a compact closed category, each wire is equipped with a direction. For any object c, a forward-pointing wire labeled c is considered equivalent to a backward-pointing wire labeled c^*, i.e. $\xrightarrow{c}$ is the same as $\xleftarrow{c^*}$. The cup and cap discussed above are in fact the unit and counit morphisms; they are drawn as follows.

In wiring diagrams, the snake equations (4.14) are then drawn as follows:



Note that the pictures in Eq. (4.13) correspond to ϵ_{sound} and η_{sound^*}.

Recall the notion of monoidal closed poset; a monoidal category can also be monoidal closed. This means that for every pair of objects $c, d \in \text{Ob}(C)$ there is an object $c \multimap d$ and an isomorphism $C(b \otimes c, d) \cong C(b, c \multimap d)$, natural in b. While we will not prove it here, compact closed categories are so-named because they are a special type of monoidal closed category.

Proposition 4.42. *If C is a compact closed category, then*

1. *C is monoidal closed;*

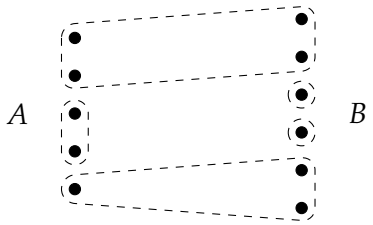
and for any object $c \in \text{Ob}(C)$,

2. *if c^* and c' are both duals to c then there is an isomorphism $c^* \cong c'$; and*
3. *there is an isomorphism between c and its double-dual, $c \cong c^{**}$.*

Before returning to co-design, we give another example of a compact closed category, called **Corel**, which we'll see again in the next two chapters.

Example 4.43. Recall, from **??**, that an equivalence relation on a set A is a reflexive, symmetric, and transitive binary relation on A. Given two finite sets, A and B, a *corelation* $A \to B$ is an equivalence relation on $A \sqcup B$.

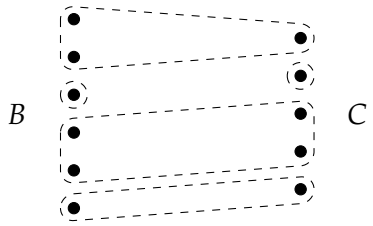
So, for example, here is a corelation from a set A having five elements to a set B having six elements; two elements are equivalent if they are encircled by the same dashed line.



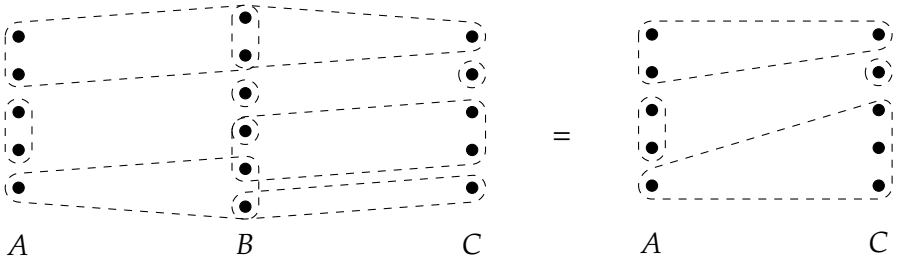
There exists a category, denoted **Corel**, where the objects are finite sets, and where a morphism from $A \to B$ is a corelation $A \to B$. The composition rule is simpler to look at that to write down formally.[2] If in addition to the corelation $\alpha \colon A \to B$ above

[2] To compose corelations $\alpha \colon A \to B$ and $\beta \colon B \to C$, we need to construct an equivalence relation $\alpha.\beta$ on $A \sqcup C$. To do so requires three steps: (i) consider α and β as relations on $A \sqcup B \sqcup C$, (ii) take the transitive closure of their union, and then (iii) restrict to an equivalence relation on $A \sqcup C$. Here is the formal description. Note that as binary relations, we have $\alpha \subseteq (A \sqcup B) \times (A \sqcup B)$, and $\beta \subseteq (B \sqcup C) \times (B \sqcup C)$.

we have another corelation $\beta\colon B \to C$



Then the composite $\beta \circ \alpha$ of our two corelations is given by



That is, two elements are equivalent in the composite corelation if we may travel from one to the other staying within equivalence classes of either α or β.

The category **Corel** may be equipped with the symmetric monoidal structure $(\varnothing, \sqcup)$. This monoidal category is compact closed, with every finite set its own dual. Indeed, note that for any finite set A there is an equivalence relation on $A \sqcup A \coloneqq \{(a,1),(a,2) \mid a \in A\}$ where each part simply consists of the two elements $(a,1)$ and $(a,2)$ for each $a \in A$. The unit on a finite set A is the corelation $\eta_A\colon \varnothing \to A \sqcup A$ specified by this equivalence relation; similarly the counit on A is the corelation $\epsilon_A\colon A \sqcup A \to \varnothing$ specifed by this same equivalence relation. ⧫

Exercise 4.44. Consider the set $\underline{3} = \{1,2,3\}$.

1. Draw a picture of the unit corelation $\varnothing \to \underline{3} \sqcup \underline{3}$.
2. Draw a picture of the counit corelation $\underline{3} \sqcup \underline{3} \to \varnothing$.
3. Check that the snake equations (4.14) hold. (Since every object is its own dual, you only need to check one of them.) ◊

4.5.2 Feas as a compact closed category

We close the chapter by returning to co-design and showing that **Feas** has a compact closed structure. This is what allows us to draw the kinds of wiring diagrams we saw in Eqs. (4.1), (4.11), and (4.12): it is what puts actual mathematics behind these pictures.

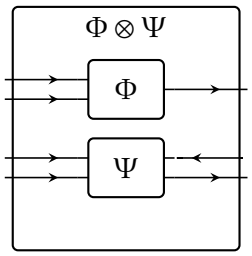
Instead of just detailing this compact closed structure for **Feas**, it's no extra work to prove that for any skeletal (commutative) quantale $(\mathcal{V}, I, \otimes)$ the profunctor category

We also have three inclusions: $\iota_{A\sqcup B}\colon A \sqcup B \to A \sqcup B \sqcup C$, $\iota_{B\sqcup C}\colon B \sqcup C \to A \sqcup B \sqcup C$, and $\iota_{A\sqcup C}\colon A \sqcup C \to A \sqcup B \sqcup C$. Recalling our notation from **??**, we define

$$\alpha.\beta \coloneqq \iota^*_{A\sqcup C}((\iota_{A\sqcup B})_!(\alpha) \vee (\iota_{B\sqcup C})_!(\beta)).$$

Prof$_{\mathcal{V}}$ is compact closed, so we'll prove this general fact. Indeed, all we need to do is construct a monoidal structure and duals for objects.

Monoidal products in Prof$_{\mathcal{V}}$ are just product categories In terms of wiring diagrams, the monoidal structure looks like stacking wires or boxes on top of one another, with no new interaction.



We take our monoidal product on **Prof**$_{\mathcal{V}}$ to be that given by the product of $\mathcal{V}$-categories; the definition was given in **??**, and we worked out several examples there. To recall, the formula for the hom-sets in $\mathcal{X} \times \mathcal{Y}$ is given by

$$(\mathcal{X} \times \mathcal{Y})((x, y), (x', y')) \coloneqq \mathcal{X}(x, x') \otimes \mathcal{Y}(y, y').$$

But monoidal products need to be given on morphisms also, and the morphisms in **Prof**$_{\mathcal{V}}$ are $\mathcal{V}$-profunctors. So given $\mathcal{V}$-profunctors $\Phi\colon \mathcal{X}_1 \nrightarrow \mathcal{X}_2$ and $\Psi\colon \mathcal{Y}_1 \nrightarrow \mathcal{Y}_2$, one defines a $\mathcal{V}$-profunctor $(\Phi \times \Psi)\colon \mathcal{X}_1 \times \mathcal{Y}_1 \nrightarrow \mathcal{X}_2 \times \mathcal{Y}_2$ by

$$(\Phi \times \Psi)((x_1, y_1), (x_2, y_2)) \coloneqq \Phi(x_1, x_2) \otimes \Psi(y_1, y_2).$$

The monoidal unit in Prof$_{\mathcal{V}}$ is 1 To define a monoidal structure on **Prof**$_{\mathcal{V}}$, we need not only a monoidal product—as defined above—but also a monoidal unit. Recall the $\mathcal{V}$-category **1**; it has one object, say 1, and $\infty(1, 1) = I$ is the monoidal unit of $\mathcal{V}$. We take **1** to be the monoidal unit of **Prof**$_{\mathcal{V}}$.

Exercise 4.45. In order for **1** to be a monoidal unit, there are supposed to be isomorphisms $\mathcal{X} \times \mathbf{1} \to \mathcal{X}$ and $\mathbf{1} \times \mathcal{X} \to \mathcal{X}$, for any $\mathcal{V}$-category $\mathcal{X}$. What are they? ◊

Duals in Prof$_{\mathcal{V}}$ are just opposite categories In order to regard **Prof**$_{\mathcal{V}}$ as a compact closed category (Definition 4.41), it remains to specify duals and the corresponding cup and cap.

Duals are easy: for every $\mathcal{V}$-category $\mathcal{X}$, its dual is its opposite category $\mathcal{X}^{\mathrm{op}}$ (see **??**). The unit and counit then look like identities. To elaborate, the unit is a $\mathcal{V}$-profunctor $\eta_{\mathcal{X}}\colon \mathbf{1} \nrightarrow \mathcal{X}^{\mathrm{op}} \times \mathcal{X}$. By definition, this is a $\mathcal{V}$-functor

$$\eta_{\mathcal{X}}\colon \mathbf{1} \times \mathcal{X}^{\mathrm{op}} \times \mathcal{X} \to \mathcal{V};$$

we define it by $\eta_{\mathcal{X}}(1, x, x') \coloneqq \mathcal{X}(x, x')$. Similarly, the counit is the profunctor $\epsilon_{\mathcal{X}}\colon (\mathcal{X} \times \mathcal{X}^{\mathrm{op}}) \nrightarrow \mathbf{1}$, defined by $\epsilon_{\mathcal{X}}(x, x', 1) \coloneqq \mathcal{X}(x, x')$.

Exercise 4.46. Check these proposed units and counits do indeed obey the snake equations Eq. (4.14). ◊

4.6 Summary and further reading

This chapter introduced three important ideas in category theory: profunctors, categorification, and monoidal categories. Let's talk about them in turn.

Profunctors generalize binary relations. In particular, we saw that the idea of profunctor over a monoidal poset gave us the additional power necessary to formalize the idea of a feasibility relation between resource posets. The idea of a feasibility relation is due to Andrea Censi; he called them *monotone codesign problems*. The basic idea is explained in [**Censi:2015a**], where he also gives a programming language to specify and solve codesign problems. In [**censi:2017a**], Censi further discusses how to use estimation to make solving codesign problems computationally efficient.

We also saw profunctors over the poset **Cost**, and how to think of these as bridges between Lawvere metric space. We referred earlier to Lawvere's paper [**Lawvere:1973a**]; plenty more on **Cost**-profunctors can be found there.

Profunctors, however are vastly more general than the two examples of have discussed; $\mathcal{V}$-profunctors can be defined not only when $\mathcal{V}$ is a poset, but for any symmetric monoidal category. A delightful, detailed exposition of profunctors and related concepts such as equipments, companions and conjoints, symmetric monoidal bicategories can be found in [**Shulman:2008a**; **Shulman:2010a**].

We have not defined symmetric monoidal bicategories, but you would be correct if you guessed this is a sort of categorification of symmetric monoidal categories. Baez and Dolan tell the subtle story of categorifying categories to get ever *higher* categories in [**Baez.Dolan:1998**]. Crane and Yetter give a number of examples of categorification in [**Crane.Yetter:1996a**].

Finally, we talked about monoidal categories and compact closed categories. Monoidal categories are a classic, central topic in category theory, and a quick introduction can be found in [**MacLane:1998a**]. Wiring diagrams play a huge role in this book and in applied category theory in general; while in informal use for years, these were first formalized in the case of monoidal categories. You can find the details here [**Joyal.Street:1993a**; **Joyal.Street.Verity:1996a**].

Compact closed categories are a special type of structured monoidal category; there are many others. For a broad introduction to the different flavors of monoidal category, detailed through their various styles of wiring diagram, see [**selinger2010survey**].