Mathematical Foundations of Engineering Co-Design

Andrea Censi*, Jonathan Lorand[1], David I. Spivak†, Joshu

*Institute for Dynamic Systems and Control (IDSC), Department of Mechanical and Process Engineering (D-MAVT), ETH Zürich, Zürich, Switzerland acensi,jlorand,gzardini@ethz.ch

†Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA.dspivak@gmail.com

‡Department of Computer Science, University of Oxford, Oxford, UK. joshua.tan@cs.ox.ac.uk

This paper is about engineering design and category theory: an odd couple, consisting of one of the most concrete and one of the most abstract fields of human knowledge. We will show how category theory can help in defining, understanding, and solving formal engineering design problems. This theory is especially well-suited for heterogeneous components with complex recursive co-design constraints.

This paper is written for two audiences: we want to explain to engineers the power and the necessity of the category-theoretical language to describe complex design problems; and we want to call the mathematicians' attention to the correspondence between very abstract category theory and extremely concrete engineering design choices that are observable in any artefact.

For the reader who already knows category theory: We introduce a category **DP** of *design problems* and study its properties. We show that **DP** is a generalization of the category **Rel** of relations. Diagrams in **DP** describe the co-design constraints among different subsystems of a large system. We show that **DP** is *compact closed*, which makes the diagrams easy to manipulate as they behave similarly to other formalisms such as signal flow diagrams. Finally, we describe the 2-category structure of **DP**, which models naturally the dominance between alternative designs.

Contents

1 Introduction

2 Concepts of Formal Engineering Design

This section introduces the basic concepts of engineering design, and describes the questions we want to answer with our work.

2.1 Basic concepts of formal engineering design

We will informally introduce some of the basic nomenclature of engineering design [**antonsson2005formal**]. Later, all these concepts will find a formal definition in the language of category theory.

Functionality and functional requirements You are an engineer in front of an empty whiteboard, ready to start designing the next product. The first question to ask is: What is the *purpose* of the product to be designed? The purpose of the product is expressed by the *functional requirements*, sometimes called *functional specifications*, or simply *function*.

Unfortunately, the word "function" conflicts with the mathematical concept. Therefore, we will talk about *functionality*. Moreover, we will never use the word "function", and instead use *map* to denote the mathematical concept.

Example 2.1. These are a few examples of functional requirements:

- A car must be able to transport at least $n \geq 4$ passengers.
- A battery must store at least 100 kJ of energy.
- An autonomous vehicle should reach at least 20 mph while guaranteeing safety.

Resources and resource constraints We call *resources* what we need to pay to realize the given functionality. In some contexts, these are better called *costs*, or *dependencies*.

Example 2.2. These are a few examples of resource constraints:

- A car should not cost more than 15,000 USD.
- A battery should not weigh more than 1 kg.
- A process should not take more than 10 s.

Duality of functionality and resources There is an interesting duality between functionality and resources. When designing systems, one is given functional requirements, as a *lower bound* on the functionality to provide, and one is given resource constraints, which are an *upper bound* on the resources to use.

As far as design objectives go, most can be understood as either *minimize resource usage* or *maximize functionality provided*.

This duality between functionality and resources will be at the center of our formalization.

Non-functional requirements Functionality and resources do not cover all the requirements— there is, for example, a large class of *non-functional requirements* [**??**] such as the extensibility and the maintainability of the system. Nevertheless, functionality and resources can express most of the requirements which can be quantitatively evaluated, at least prior to designing, assembling, and testing the entire system.

Implementation space The *implementation space* or *design space* is the set of all possible design choices that could be chosen; by *implementation*, or the word "design", used as a noun, we mean one particular set of choices. The implementation space I is the set over which we are optimizing; an implementation $i \in I$ is a particular point in that set (Fig. 2.1).

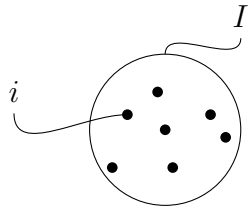


Figure 2.1: An *implementation* i is a particular point in the implementation space I.

The interconnection between functionality, resources, and implementation spaces is as follows. We will assume that, given one implementation, we can evaluate it to know the functionality and the resources spaces (Fig. 2.2).

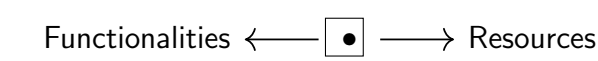


Figure 2.2: Evaluation of specific implementations to get functionality and resources spaces.

Systems and components Engineers tend to approach complex design problems hierarchically. The usual nomenclature refers to *systems* being decomposed into *components*.

What is a system, and what is a component? Here is a great quote:

> A system is composed of components;
> a component is something you understand.
> —Howard Aiken[1]

The first part of the quote, "A system is composed of components", is plain as day as much as it is tautological. We could equally say: "A system is partitioned in parts".

The second part, "a component is something you understand", is the insightful part: We call "system" what is not obvious to understand, even if we understand all its components separately.

Whether something can be considered a component, an indivisible atom in the theory, depends on the context.

Interfaces and interconnection Components are *interconnected* to create a system. This implies that we have defined the *interfaces* of components, which have the dual function of delimiting when one component ends and another begins, and also to describe exactly what is the nature of their interaction.

In this paper, we will present a formalism in which the functionality and resources are the interfaces used for interconnection: Two components are

[1] Howard H. Aiken (1900-1973), creator of the MARK I computer. Quoted by Kenneth E. Iverson (1920-2004), creator of programming language APL. Quoted in this paper[**that-paper**], but ultimately sourceless and probably apocryphal.

connected if the resources required by the first correspond to the functionality provided by the second.

Abstraction By *abstraction*, one usually means that it is possible to "zoom out", in the sense that a system of components can be seen as a component itself, which can be part of larger systems.

Compositionality A *compositional* property is a property that is preserved by interconnection and abstraction; assuming each component in a system satisfies that property, also the system as a whole satisfies the property.

Example 2.3. One can compose two electronic circuits by joining their terminals to obtain another electronic circuit. We would say that the property of being an electronic circuit is compositional.

2.2 Queries in design

Design Queries The design queries we will present throughout this paper are the following:

- Given a certain functionality, find the minimal resources that can realize it, or provide a proof that there are none.
- Given certain resources, find the maximal functionality that can be realized, or provide a proof that there are none.

3 Thinking about how things transform into each other

3.1 Category theory helps reasoning about interfaces and transformations

We argue that all the concepts of formal engineering design discussed in the previous section can be naturally described in the language of category theory. We will start with a simple example. In an electric car, electric power is turned by the engine into rotational motion of the axle, and then the rotational motion is converted into translational motion by the wheels and their friction with the road. In this simple setting, we can identify the key systems and subsystems: car, engine, axle, wheels, and road.

We can identify the functionality/resources of interest as: electric power, rotational motion, and translational motion. Note that each of these quantities plays a dual role. For example, the rotational motion is something which is produced by the motor, so it is a functionality for the motor, while it is a resource for axle/wheels, because they need it to provide translational motio

For a first qualitative description of the scenario, we might choose to just keep track of what functionality/resource is converted into what. For this qualitative description, we can draw a diagram in which each resource is a point (Fig. 3.1).

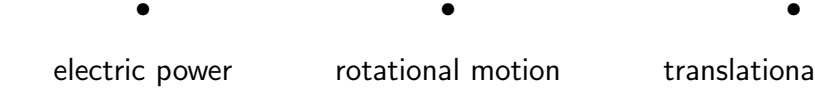


Figure 3.1: Resources and functionalities in the electric car example.

Furthermore, we can draw an arrow between two resources if we can obtain one from the other. In the example, we have described how electric power becomes rotational motion, described by the engine arrow, and how rotational moti becomes translational motion, described by the wheel arrow (Fig. 3.2).

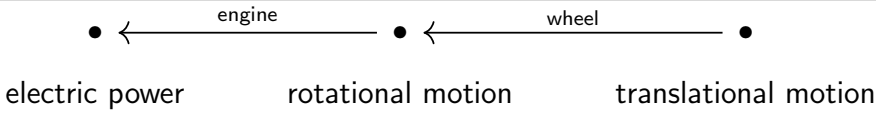


Figure 3.2: System components connect resources and functionalities.

In this representation, the arrows are the components of the system. We will learn how to compose these arrows according to the rules of category theory.

If we use the semantics that an arrow from resource X to resource Y means "having Y is enough to obtain X", then, since Y is enough for Y per definition, we can add a self-loop for each resource. We will call the self-loops *identities* (Fig. 3.3).

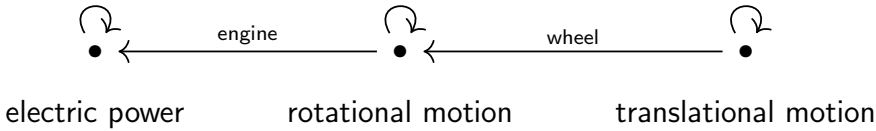


Figure 3.3: System components and identities.

Furthermore, we might consider the idea of composition of arrows. If from Y wecan get X and from Z we can get Y, then from Z we can get X. In our example, if the arrows wheel and engine exist, then also the arrow "engine then wheel" exists (Fig. 3.4).

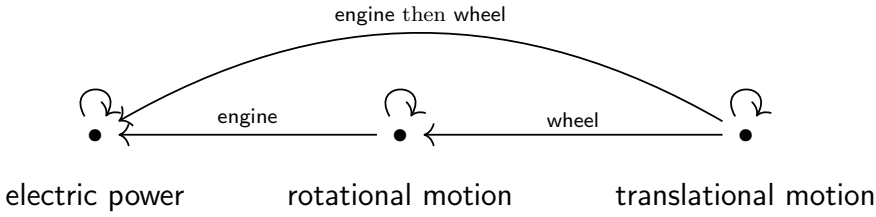


Figure 3.4: Composition of system components .

GZ: Figure has to go here

A "category" is an abstract mathematical structure which captures the intuitive notions of objects and arrows between them. The following is the formal definition.

Definition 3.1 (Category)**.** A *category* $\mathbf{C}$ is specified by four components:

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

Furthermore, constituents are required to satisfy the following conditions:

1. *Unitality*: for any morphism $f \in \mathrm{Hom}(X, Y)$,
$$\mathrm{id}_X \fatsemi f = f = f \fatsemi \mathrm{id}_Y . \tag{3.1}$$
2. *Associativity*: for $f \in \mathrm{Hom}_{\mathbf{C}}(X, Y)$, $g \in \mathrm{Hom}_{\mathbf{C}}(Y, Z)$, and $h \in \mathrm{Hom}_{\mathbf{C}}(Z, W)$,
$$(f \fatsemi g) \fatsemi h = f \fatsemi (g \fatsemi h). \tag{3.2}$$

Remark 3.2. The set of morphisms $\mathbf{C}(X, Y)$ is sometimes denoted $\mathrm{Hom}_{\mathbf{C}}(X, Y)$ and called the "hom-set from X to Y". The "Hom" comes from the word "homomorphism".

Remark 3.3 (Composition notation). We denote composition of morphisms in a somewhat unusual way–sometimes preferred by category-theorists and computer scientists–namely in *diagrammatic order*. That is, given $f\colon A \to B$ and $g\colon B \to C$, we denote their composite by $(f \fatsemi g)\colon A \to C$, pronounced "$f$ then g". This is in contrast to the more typical notation for composition, namely $g \circ f$, or simply gf, which reads as "g after f". The notation $f \fatsemi g$ is sometimes called *infix notation*.

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

- We do not need to draw the identity arrows from one object to itself, because, by Definition 3.1, they always exist.

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

- Given arrows $A \to B$ and $B \to C$, we do not need to draw their composition because, by Definition 3.1, this composition is guaranteed to exist.

With these conventions, we can just draw the arrows engine and wheel in the diagram, and the rest of the diagram is implied (Fig. 3.5).

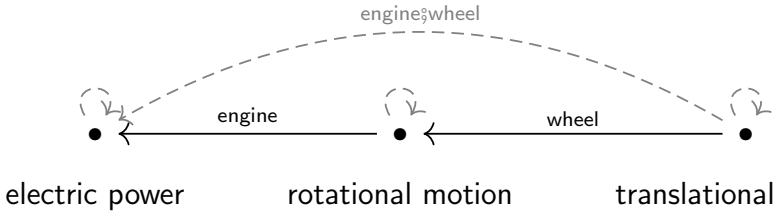


Figure 3.5: Electric car example. The grey arrows are implied by the properties of a category.

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

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

We can slightly expand this example by noting the inverse transformations. In an electric car it is possible to regenerate power; that is, we can obtain rotational motion of the wheels from translational motion (via the morphism move), and then convert the rotational motion into electric power (via the morphism dynamo) (Fig. 3.6, Fig. 3.7).

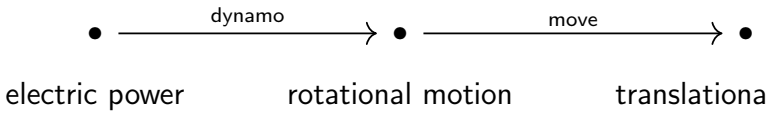


Figure 3.6

GZ: Conflict in composition order in figures, discuss on how to fix Given the semantics of the arrows in a category, all compositions of arrows exist, even if they are not drawn explicitly. For example, we can consider the composition $f \fatsemi g \fatsemi k \fatsemi h$, which converts electric power into rotational motion, into translational motion, then back to rotational motion and electric power. Note that this is an arrow that has the same head and tail as the identity

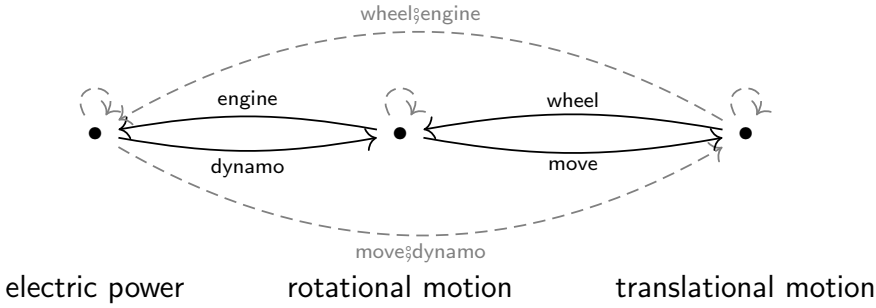


Figure 3.7

arrow on electric power (Fig. 3.8). However, these two arrows are not necessarily the same. In this example we are representing physical systems, so we would in fact not expect them to be the same, since there will be some losses during the many conversions.

Figure 3.8: There can be multiple morphisms from an object to itself.

It is important to recognize that there might also be two or more distinct arrows between two distinct resources (Fig. 3.9). Each arrow between X and Y represents a different way to obtain X from Y. This is one of the aspects which will make category theory powerful enough to reason about design.

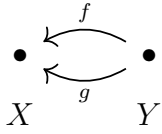
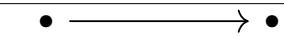


Figure 3.9

The directionality of the arrows is also important. While the convention of which resource is the tail and which the head is just a typographic convention, it might be the case that we know how to convert one resource into another, but not vice versa. Fig. 3.10 shows an example of a diagram that describes a process which is definitely not invertible.



electric power heat

Figure 3.10: An example of a process which is not invertible.

3.2 Currency categories

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

We start with a set C of labels for all the currencies we wish to consider, i.e.

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

For each currency $\mathsf{c} \in \mathsf{C}$ we define an object $\mathbb{R} \times \{c\}$ which represents possible amounts of the given currency c (we will ignore the issue of rounding currencies to an accuracy of two decimals places, and we allow negative amounts). The currency label keeps track of which "units" we are using.

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

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

by the formula

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

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

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

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

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

$$E_{aa',a'b+b'} \colon \mathbb{R} \times \{\mathsf{c}_1\} \to \mathbb{R} \times \{\mathsf{c}_3\}. \tag{3.3}$$

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

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

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

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

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

Remark 3.4. In the above specification of our category of currency exchangers, we can actually just work with the set of currency labels C as our objects, instead of using "amounts" of the form $\mathbb{R} \times \{\mathsf{c}\}$ as our objects. Indeed, on a mathematical level, the definition of currency exchangers and their composition law (Eq. (3.3)) do not depend on using amounts! Namely, a currency exchanger $E_{a,b}$ is specified by the pair of numbers $\langle a, b \rangle$, and the composition law (3.3) may then, in this notation, be written as

$$\langle a, b \rangle \fatsemi \left\langle a', b' \right\rangle = \left\langle a'a, a'b + b' \right\rangle. \tag{3.4}$$

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

Following the above remark:

Definition 3.5 (Category **Curr**)**.** A *category of currencies* **Curr** is specified by:

1. *Objects:* a collection of currencies C.
2. *Morphisms:* given two currencies $\mathsf{c}_1, \mathsf{c}_2 \in \mathsf{C}$, morphisms between them are currency exchangers $\langle a, b \rangle$ from c_1 to c_2.
3. *Identity morphism:* given an object $\mathsf{c} \in \mathsf{C}$, its identity morphism is the currency exchanger $\langle 1, 0 \rangle$. We also call such morphisms "trivial currency exchangers".

4. *Composition of morphisms:* the composition of morphisms is given by the formula (3.4).

As an illustration, consider three currency exchange companies ExchATM, Money and Frankurrencies, which operate on several currencies (Table 6.1).

Company name	Exchanger label	Direction	a (exchange rate)	b (fixe
ExchATM	A	USD to CHF	$0.95\,^{\text{CHF}}/_{\text{USD}}$	
ExchATM	B	CHF to USD	$1.05\,^{\text{USD}}/_{\text{CHF}}$	
ExchATM	C	USD to SGD	$1.40\,^{\text{SGD}}/_{\text{USD}}$	
MoneyLah	D	USD to CHF	$1.00\,^{\text{CHF}}/_{\text{USD}}$	
MoneyLah	E	SGD to USD	$0.72\,^{\text{USD}}/_{\text{SGD}}$	
Frankurrencies	F	EUR to CHF	$1.20\,^{\text{CHF}}/_{\text{EUR}}$	
Frankurrencies	G	CHF to EUR	$1.00\,^{\text{EUR}}/_{\text{CHF}}$	

Table 3.1: Three currency exchange companies operating different currencies.

We can represent this information as a graph, where the nodes are the currencies and the edges are particular exchange operations (Fig. 3.11).

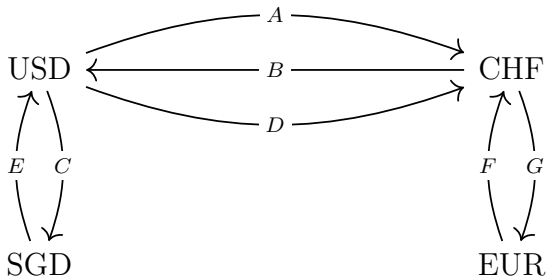


Figure 3.11: Three currency exchange companies operating different currencies as a graph.

There is a currency category built from the information in Table 6.1 and the graph in Fig. 3.11. Its collection of objects is the set {EUR, USD, CHF, SGD} and it morphisms are, in total,

- the trivial currency exchanger (identity morphism) $\langle 1, 0 \rangle$ for each of the four currencies (which are the objects),
- the currency exchangers corresponding to each item in Table 6.1,
- all possible compositions of the currency exchangers listed in Table 6.1.

The phrase "all possible compositions" is a bit vague. What we mean here can be made more precise. It corresponds to a general recipe for starting with a graph G, such as in Fig. 3.11, and obtaining from it an associated category, called the *free category on* G. We introduce this concept in the next section.

JL: The following paragraph(s) might be best moved to a later part in the text, once pareto fronts and optimization have been discussed a little bit. The category **Curr** represents the set of all possible currency exchangers that could ever exist. However, in this set there would be very irrational agents. For example, there is a currency exchanger that given 1 USD, will give you back 2 USD, or even will not give you back any money. This highlights a recurring topic: often mathematicians will be happy to define a broader category of objects, while, in practice, the engineer will find herself thinking about a more constrained set of objects. In particular, one would like to find the best conversions. These can be expressed as *Pareto fronts*. To do so, one only needs to iterate a finite number of times, since the optimal path (conversion morphism), if such a morphism exists, will never pass through the same currency more than once. JL: the previous sentence needs explaining This is valid if we assume the action of converting back and forth the same currency to have a cost (through the commission) higher than 0. To see this, consider three currencies $\mathsf{A}, \mathsf{B}, \mathsf{C}$, a currency exchanger $\langle a, b\rangle$ from A to B , a currency exchanger $\langle c, d\rangle$ from B to C , and a currency exchanger $\langle e, f\rangle$ from C to A. JL: more pedagogical The composition of the currency exchangers reads:

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

Assuming $e = a^{-1}$ (i.e., an exchange rate direction is not more profitable than the other), and $h \neq 0$, because of the commissions one can show that there are multiple morphisms from A to A, and that the identity morphism is the most "convenient" one.

3.3 Generating categories from graphs

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

Definition 3.6 (Graph)**.** A (directed) *graph* $G = \langle V, A, s, t\rangle$ consists of a set of vertices V, a set of arrows A, and two functions $s, t\colon A \to V$, called

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

Remark 3.7. Both directed graphs and undirected graphs play a prominent role in many kinds of mathematics. In this text, we work primarily with directed graphs and so, from now on, we will drop the "directed": unless indicated otherwise, the word "graph" will mean "directed graph".

Definition 3.8 (Paths)**.** Let G be a graph. A *path* in G is a sequence of arrows such that the target of one arrow is the source of the next. The *length* of a path is the number of arrows in the sequence. We also formally allow for sequences made up of "zero-many" arrows (such paths therefore have length zero). We call such paths *trivial* or *empty*. If paths describe a journey, then trivial paths correspond to "not going anywhere". The notions of source and target for arrows extend, in an obvious manner, to paths. For trivial paths, the source and target always coincide.

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

Definition 3.9 (Free category on a graph)**.** Let $G = (V, A, s, t)$ be a graph. The *free category on* G, denoted $\mathbf{Free}(G)$, has as objects the vertices V of G, and given vertices $x \in V$ and $y \in V$, the morphisms $\mathbf{Free}(G)(x, y)$ are the paths from x to y. The composition of morphisms is given by concatenation of paths, and for any object $x \in V$, the associated identity morphism id_x is the trivial path which starts and ends at x.

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

Exercise 3.10. Consider the following five graphs. For each graph G, how many morphisms in total are there in the associated category $\mathbf{Free}(G)$?

4 Thinking about how things connect to each other

5 Thinking about how things connect to each other

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

5.1 Trekking

Consider a geographical region whose locations are expressed through coordinates $(x, y) \in \mathbb{R}^2$, e.g. as given by a map of that region. Furthermore, consider a function $\mathsf{alt} : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ which, for a known location, returns its altitude.

We can think about this situation using a category, call it **trek**, where objects are geographical locations $\langle x, y \rangle \in \mathbb{R}^2$ and morphisms are continuous paths between them. The identity morphism for each location consists of the trivial path (i.e., not moving), and composition is given by concatenation of paths. JL: We need to be more precise about what "continuous path" means here! The typical mathematical definition of paths from topology is as a function of a ("time") parameter, and leads to a well-known situation where concatenation is not an associative operation on the nose... and/or there is also issue that there are crazy kinds of continuous paths, such as space-filling curves... perhaps this example can be modified a bit to capture the basic idea, but avoid the math issues... GZ: Agree, here we just want the connectivity and the filtering of paths which have too large inclinations

Suppose that a human can only traverse trails which have a maximum inclination of $\alpha > 0$ when going uphill and $\beta > 0$ when going downhill. We can now think of the aforementioned human, wanting to go from a location $\langle x, y \rangle$ to a location $\langle v, w \rangle$. Finding a path consists of finding at least a morphism in $\mathrm{Hom}_{\mathbf{trek}}(\langle x, y \rangle, \langle v, w \rangle)$ satisfying the condition on the maximum inclinations α and β.

Using the terminology from Section 3.3, we can see that **trek** is the free category on a graph with vertices given by geographical locations $\langle x, y\rangle \in \mathbb{R}^2$ and arrows given by paths between them. In particular, a valid path $p\colon \langle x, y\rangle \mapsto \langle v, w\rangle$ for the human to be able to reach a destination, has not to exceed the maximum inclination α when climbin and the maximum inclination β when descending.

5.2 Mobility

For a specific mode of transportation, say a car, we can define a graph $G_\mathrm{c} = \langle V_\mathrm{c}, A_\mathrm{c}, s_\mathrm{c}, t_\mathrm{c}\rangle$, where V_c represents geographical locations which the car can reach and A_c represents the paths it can take (e.g. roads). Similarly, we consider a graph $G_\mathrm{s} = (V_\mathrm{s}, A_\mathrm{s}, s_\mathrm{s}, t_\mathrm{s})$, representing the subway system of a city, with stations V_s and subway lines going through paths A_s, and a graph $G_\mathrm{b} = \langle V_\mathrm{b}, A_\mathrm{b}, s_\mathrm{b}, t_\mathrm{b}\rangle$, representing onboarding and offboarding at airports. In the following, we want to express intermodality: the phenomenon that someone might travel to a certain intermediate location in a car and then take the subway to reach their final destination.

By considering the graph $G = (V, A, s, t)$, with $V = V_\mathrm{c} \cup V_\mathrm{s} \cup V_\mathrm{b}$ and $A = A_\mathrm{c} \cup A_\mathrm{s} \cup A_\mathrm{b}$, we obtain the desired intermodality graph. Graph G can be seen as a new category, with objects V and morphisms A.

Example 5.1. Consider the **car** category, describing your road trip in California, with

$$V_\mathrm{c} = \{\mathsf{SFO}_\mathrm{c}, \mathsf{S.\ Mateo}, \mathsf{Half\ Moon\ Bay}, \mathsf{SBP}_\mathrm{c}, \mathsf{Lake\ Balboa}, \mathsf{LAX}_\mathrm{c}\},$$

and arrows as in Fig. 5.1. The nodes represent typical touristic road trip checkpoints in California and the arrows famous highways connecting them.

$$\mathsf{SFO}_\mathrm{c} \xrightarrow{\text{US101}} \mathsf{S.\ Mateo} \xrightarrow{\text{CA92}} \mathsf{H.\ M.\ Bay} \xrightarrow{\text{CA1}} \mathsf{SBP}_\mathrm{c} \xrightarrow{\text{US101S}} \mathsf{Lake\ Balboa} \xrightarrow{\text{I405}} \mathsf{LAX}_\mathrm{c}$$

Figure 5.1: The **car** category.

Furthermore, consider the **flight** category with $V_\mathrm{f} = \{\mathsf{SFO}_\mathrm{f}, \mathsf{SJC}, \mathsf{SBP}_\mathrm{f}, \mathsf{LAX}_\mathrm{f}\}$ and arrows as in Fig. 5.2. The nodes represent airports in california and the arrows represent connections, offerend by specific flight companies.

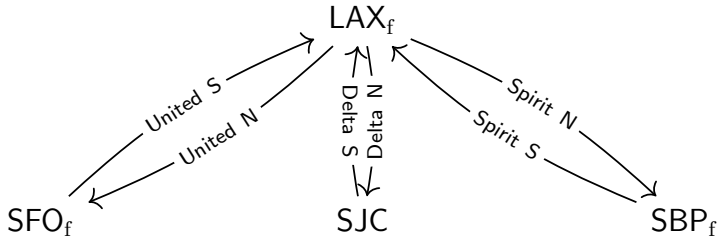


Figure 5.2: The **flight** category.

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

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

and arrows as in Fig. 5.3. Nodes represent airports and airport parkings, and arrows represent the onboarding and offboarding paths one has to walk to get from the parkings to the airport and vice-versa.

Figure 5.3: The **board** category.

The combination of the three, which we call the *intermodal graph*, can be represented as a graph, with red arrows for the car network, blue arrows for the flight network, green arrows for the boarding network, and black dashed arrows for intermodal morphisms, arising from composition of morphisms involving multiple modes (Fig. 5.4). Imagine that you are in the parking of LAX airport and you want to reach S. Mateo. From there, you will e.g. onboard to a United flight to SFO_f, will then offboard reaching the parking lot SFO_c, and drive on highway $\mathsf{US-101}$ reaching S. Mateo. This is intermodality.

The intermodal network category **intermodal** is the free category on the graph illustrated in Fig. 5.4.

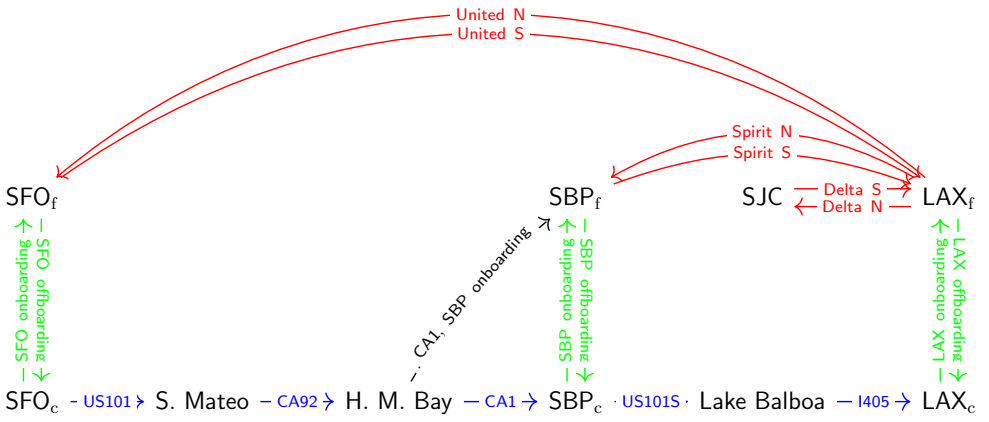


Figure 5.4: Intermodal graph. The dashed arrows represent intermodal morphisms, and we depict just one of them for simplicity.

GZ: Maybe not needed? If so, rewrite JL: I think this is nice to include! But then maybe for clarity we would want to actually remove the dashed arrow in Figure 18, since that is not part of the graph we are thinking to take the free category of. Technically we could also leave it in, but that seems to confuse the situation, no?

6 Thinking about attributes and sameness

JL: Maybe we could make this its own chapter about "attributes and constraints" and cook up some set-based examples, and then use those as motivation to talk about the category of sets and the notion of sub-category... perhaps some examples related to David Spivak's database framework could be useful here.... below is a first attempt/draft of this.

6.1 Sets, functions, databases

Suppose we want to buy an electric stepper motor for a robot that we are building, and for this we consult a catalogue of electric stepper motors[1]. The catalogue might be organized as a large table, where on the left-hand side there is a column listing all available motors (identified with a model name), and the remaining columns correspond to different attributes that each of the models of motor might have, such as the name of the company that manufactures the motor, the size dimensions, the weight, the maximum power, the price, etc. A simple illustration is provided in Table 6.1.

Motor ID	Company	Size [mm^3]	Weight [g]	Max Power [W]	Cost [USD]
NEMA1204	SOYO	20 x 20 x 30	60.0	2.34	19.95
NEMA1206	SOYO	28 x 28 x 45	140.0	3.00	19.95
NEMA1207	SOYO	35 x 35 x 26	130.0	2.07	12.95
NEMA2267	SOYO	42 x 42 x 38	285.0	4.76	16.95
NEMA2279	Sanyo Denki	42 x 42 x 31.5	165.0	5.40	164.95
NEMA1478	SOYO	56.4 x 56.4 x 76	1,000	8.96	49.95
NEMA2299	Sanyo Denki	50 x 50 x 16	150.0	5.90	59.95

Table 6.1: A simplified catalogue of motors.

One useful way to think of tables of data is in terms of sets and functions.

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

In this example, we can consider the set

$M \coloneqq \{\mathsf{NEMA1204}, \mathsf{NEMA1206}, \mathsf{NEMA1207}, \mathsf{NEMA2267}, \mathsf{NEMA2279}, \mathsf{NEMA1478},$

of models of motors, as well as the set $C \coloneqq \{\mathsf{SOYO}, \mathsf{Sanyo\ Denki}\}$ of manufacturing companies, the set S of possible motor sizes, the set W of possible weights, the set J of possible maximal powers, and the set P of possible prices. Each attribute of a motor may be thought of as a function from the set M to set of possible values for the given attribute. For example, there is a function $\mathsf{Company}\colon M \to C$ which maps each model to the corresponding company that manufactures it. So, according to Table 6.1, we have e.g. $\mathsf{Company}(\mathsf{NEMA1204}) = \mathsf{SOYO}$, and $\mathsf{Company}(\mathsf{NEMA2279}) = \mathsf{Sanyo\ Denki}$, etc.

Note that in "real life", the catalogue of motors might not have seven entries, as in Table 6.1, but has in fact hundreds of entries, and is implemented digitally as a database, i.e. a collection of interrelated tables. In this case, we will want to be able to search and filter the data based on various criteria. Many natural operations on tables and databases may be described simply in terms of operations with functions. We will use this setting as a way to introduce compositional aspects of working with sets and functions, and a preview of how this might be useful for thinking, in particular, about databases.

Sticking with Table 6.1, suppose, for instance, that we want to consider only motors from Company Sanyo Denki. In terms of the function

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

this corresponds to the preimage $\mathsf{Company}^{-1}(\{\mathsf{Sanyo\ Denki}\}) = \{\mathsf{NEMA2279}, \mathsf{NEI}$ which is a subset of the set M. Or, we may want to consider only motors which cost between 40 and 200 USD. In terms of the obvious function

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

this means we wish to restrict ourselves to the preimage

$$\mathsf{Price}^{-1}(\{49.95, 59.95, 164.95\}) = \{\mathsf{NEMA1478}, \mathsf{NEMA2299}, \mathsf{NEMA2279}\} \subseteq M.$$

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

$$\mathsf{Multiply}\colon S \to V$$
$$\langle l, w, h \rangle \mapsto l \cdot w \cdot h,$$

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

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

to obtain a function

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

which defines a new column in our table. The composition of functions is usually written as $\mathsf{Volume} = \mathsf{Multiply} \circ \mathsf{Size}$, however we stick to our convention of writing $\mathsf{Volume} = \mathsf{Size} \fatsemi \mathsf{Multiply}$. Schematically, we can represent what we did as a diagram (Fig. 6.1).

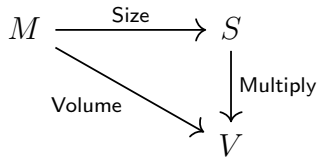


Figure 6.1: A diagram of functions.

We can interpret the arrows in this diagram as being part of a category, one where M, S, and V are among the objects, and where the functions Size, Multiply and Volume are morphisms. We probably want to consider the other sets associated with our database as also part of this category, and the other functions which we defined so far, too. One idea might be to just include all the sets and functions that we've defined so far, as well as all possible compositions of those functions, and obtain a category (maybe call it Database?) in a way that is similar to how one can build a category from a graph (Section 3.3). This would be an option. However, we may want soon to add new sets and functions to our database framework, or think about new kinds of functions between them that we had not considered before. And we might not want to re-think each time precisely which category we are working with.

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

Definition 6.1 (Category of sets)**.** The category of sets **Set** is defined by:

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

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

Definition 6.2 (Subcategory)**.** Given a category $\mathbf{C}$, a *subcategory* $\mathbf{B}$ consists of a subcollection of the collection of objects and morphisms of $\mathbf{C}$ such that:

(i) If a morphism $f\colon x \to y$ is in $\mathbf{B}$, then so are the objects x and y.
(ii) If the morphisms $f\colon x \to y$ and $g\colon y \to z$ are in $\mathbf{B}$, then so is their composite $f \fatsemi g\colon x \to z$.
(iii) If x is in $\mathbf{B}$, then so is the identity morphism id_x.

6.2 Sameness in category theory

One nice thing about the category of sets is that we are all used to working with sets and functions. And many concepts that are familiar in the setting of sets and functions can actually be reformulated in a way which makes sense for lots of other categories, if not for all categories. It can be

fun, and insightful, to see known definitions transformed into "category theory language". For example: the notion of a bijective function is a familiar concept. There are least two ways of saying what it means for a function $f\colon X \to Y$ of sets to be bijective:

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

efinition 2: "$f\colon X \to Y$ is bijective if there exists a function $g\colon Y \to X$ such that $f \fatsemi g = id_X$ and $g \fatsemi f = id_Y$".

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

Definition 6.3 (Isomorphism)**.** Let $\mathbf{C}$ be a category, let $X \in \mathbf{C}$ and $Y \in \mathbf{C}$ be objects, and let $f\colon X \to Y$ be a morphism. We say that f is an **isomorphism** if there exists a morphism $g\colon Y \to X$ such that $f \fatsemi g = \mathrm{id}_X$ and $g \fatsemi f = \mathrm{id}_Y$.

Remark 6.4. The morphism g in the above definition is called the **inverse** of f. Because of the symmetry in how the definition is formulated, it is easy to see that g is necessarily also an isomorphism, and its inverse is f.

Exercise 6.5. In Remark 6.4 we wrote *the* inverse. We do this because inverses are in fact unique. Can you prove this? That is, show that if $f\colon X \to Y$ is an isomorphism, and if $g_1\colon Y \to X$ and $g_2\colon Y \to X$ are morphisms such that $f \fatsemi g_1 = \mathrm{id}_X$ and $g_1 \fatsemi f = \mathrm{id}_Y$, and $f \fatsemi g_2 = \mathrm{id}_X$ and $g_2 \fatsemi f = \mathrm{id}_Y$, then necessarily $g_1 = g_2$.

Definition 6.6 (Isomorphic)**.** Let $\mathbf{C}$ be a category, and let $X \in \mathbf{C}$ and $Y \in \mathbf{C}$ be objects. We say that X and Y are **isomorphic** if there exists an isomorphism $X \to Y$ or $Y \to X$.

For the formulation of the definition of "isomorphic", mathematicians might often only require the existence of an isomorphism $X \to Y$, say, since by Remark 6.4 we know there is then necessarily also an isomorphism in the opposing direction, namely the inverse. We choose here the longer, perhaps more cumbersome formulation just to emphasis the symmetry of the term "isomorphic". Also note that the definition leaves unspecified whether there might be just one or perhaps many isomorphisms $X \to Y$.

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

Example 6.7 (Sizes). Suppose we are a manufacturer and we are counting how many wheels are in a certain warehouse. If W denotes the set of wheels that we have, then counting can be modelled as a function $f\colon W \to \mathbb{N}$ to the natural numbers. If we find that there are, say, 273 wheels, then our counting procedure gives us a bijective function from W to the set $\{1, 2, 3, ...272, 273\}$. In this case, we don't care which specific wheel we counted first, second, or last. We could just as well have counted in a different order, which would amount to a different function $f'\colon W \to \mathbb{N}$. The only thing we care about is the fact that the sets W and $\{1, 2, 3, ...272, 273\}$ are *isomorphic*; we don't need to keep track of which counting isomorphism exhibits this fact.

Example 6.8 (Relabelling). Consider the little catalogue in Table 6.1. Suppose that your old way of listing models of motors has become outdated and you need to change to a new system, where each model is identified, say, by a unique numerical 10-digit code. Relabelling each of the models with its numerical code corresponds to an isomorphism, say relabel, from the new set N of numerical codes to the old set M of model names. In contrast to the previous example, however, it is of course absolutely necessary to keep track of the isomorphism relabel that defines the relabelling. This is what holds the information of which code denotes which model.

Note also that all the other labelling functionalities in our example database

may be updated by precomposing with relabel. For example, the old "Company" label was described by a function

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

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

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

Example 6.9 (Semantic coherence). Suppose Francesca and Gabriel want to share a dish at a restaurant. Francesca only speaks Italian, and Gabriel only speaks German. Let M denote the set of dishes on the menu. For each dish, Francesca can say if she is willing to eat it, or not. This can be modeled by a function $f\colon M \to \{\text{Si, No}\}$ which maps a given dish $m \in M$ to the statement "Si" (yes, I'd eat it) or "No" (no, I wouldn't eat it). Gabriel can do similarly, and this can be modeled as a function $g\colon M \to \{\text{Ja, Nein}\}$. Then, the subset of dishes of M that both Francesca and Gabriel are willing to eat (and thus able to share) is

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

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

$$\{\text{Si, No}\} \longleftrightarrow \{♥, ☠\} \longleftrightarrow \{\text{Ja, Nein}\}$$

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

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

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

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

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

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

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

7 Thinking about resource composition

7.1 Products

Is "$X \times Y$" the same as "$Y \times X$"? It depends on the context. Intuitively, for the categories of resources treated in this work, we would not make a distinction between "having X and Y" and "having Y and X". However, there are contexts in which this is not valid. For example, if we are using $X \times Y$ to mean that we will have the resource X today, and the resource Y tomorrow, then $Y \times X$ would not be the same as $X \times Y$.

In the contexts in which this symmetry holds, we call the category "symmetric monoidal": we will talk about this more in detail in **??**.

The word "symmetric" is well-suited, because to say that the two objects are equivalent, or "isomorphic", we can postulate that we always have a way to get "$X \times Y$" starting from "$Y \times X$" and viceversa. Diagrammatically, this is depicted by the two arrows that connect them (Fig. 7.1).

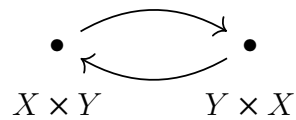


Figure 7.1: Objects equivalence in the symmetric case.

Consider the following example. To generate motion, one needs *both* an engine *and* some fuel.

We can consider motion, engine, and fuel as resources and draw them as points (Fig. 7.2).

However, it is not clear how to represent the arrow corresponding to fuel + engine = motion, because arrows only have one head and one tail. JL: Perhaps this particular example would be confusing to the reader, since earlier "engine" was used as a morphism, while here it is being used as

Figure 7.2: Resources for motion generation.

an object (??) GZ: I don't think so, instead I think it is cool to see that this way of thinking allows multiple abstraction levels One clean way to formalize this is to expand the objects in the category, by postulating that, if two resources X and Y exist, then there exists also the resource "X *and* Y".

In our example, we can postulate the existence of an object engine **and** fuel and then draw the arrow from it to motion (Fig. 7.3).

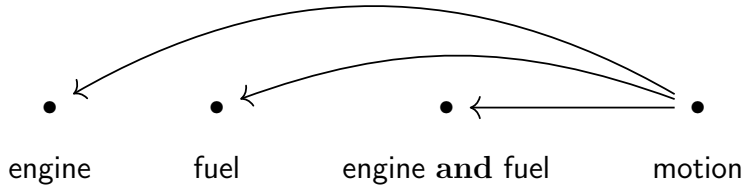


Figure 7.3: Engine **and** fuel can generate motion.

We call this construction "product", because it is equivalent to the Cartesian product of sets.

Definition 7.1 (Cartesian product of sets)**.** Given two sets A, B, their *cartesian product* is denoted $A \times B$ and defined as

$$A \times B = \{\langle a, b\rangle \mid a \in A \text{ and } b \in B\}. \tag{7.1}$$

Example 7.2. Consider the sets $\{\diamond, \star\}$ and $\{\dagger, *\}$. Their product can be represented as in Fig. 7.4.

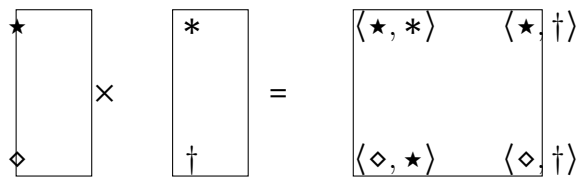


Figure 7.4: Example of cartesian product of two sets.

Given the Cartesian product of two sets, we can define two *projection maps* π_1 and π_2, which, given an element of the product, will return the first or the second element, respectively:

$$\begin{aligned} \pi_1(\langle a, b \rangle) &\coloneqq a, \\ \pi_2(\langle a, b \rangle) &\coloneqq b. \end{aligned} \tag{7.2}$$

In our case, we would be able to say that, given both $\mathsf{resource}_1$ and $\mathsf{resource}_2$ together, we can recover $\mathsf{resource}_1$ and $\mathsf{resource}_2$ separately (Fig. 7.5).

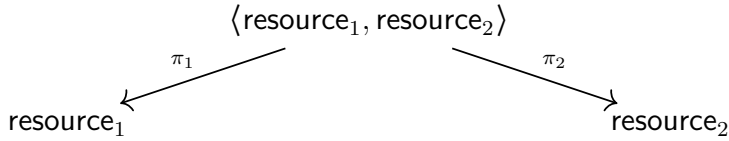
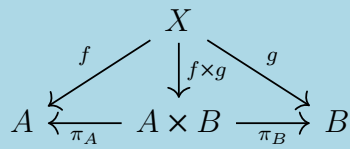


Figure 7.5: Two projection maps.

Later, we will see other examples of products, such as products of partially ordered sets. All these products are an instance of a more general concept of product that exists in category theory.

Definition 7.3 (Categorical Product)**.** Let $\mathbf{C}$ be a category and let $A, B \in \mathbf{C}$ be objects. The *product* of A and B is an object $A \times B \in \mathbf{C}$ together with *projection morphisms* $\pi_A \colon A \times B \to A$ and $\pi_B \colon A \times B \to B$, such that, given any $X \in \mathbf{C}$ and morphisms $f \colon X \to A, g \colon X \to B$, there exists a *unique* morphism $(f \times g) \colon X \to A \times B$ such that $f = (f \times g) \fatsemi \pi_A$ and $g = (f \times g) \fatsemi \pi_B$. Diagrammatically:



(7.3)

The formulation, slightly technical, hints at uniqueness properties or "universality". The definition implies that if a product exists, then it is unique, in the sense that all the admissible products can be derived from each other. Therefore, when presented with a new category, one can guess what construction might be the product, and check the definition to make sure that it is indeed *the* product.

GZ: JL: need to reformulate this last part

8 Thinking about alternatives

8.1 Coproduct

There exists a dual notion to "product" that is called "coproduct". Suppose that we are considering a hybrid car that contains two engines: an electric engine and an internal combustion engine. Both can produce motion, but each from a different source of energy. The electric engine uses electric energy; the internal combustion engine uses gasoline. The situation is as in Fig. 8.1.

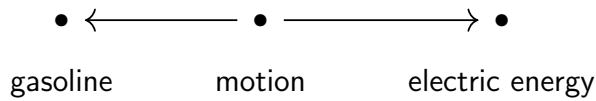


Figure 8.1: Alternative ways to generate motion.

From this we would like to conclude that we can obtain motion from *either* gasoline *or* electric energy (Fig. 8.2).

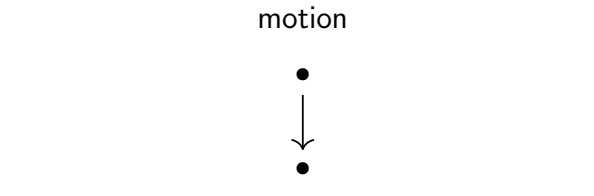


Figure 8.2: We can generate motion from either gasoline or electric energy.

To define the idea of "*either* gasoline or electric energy" we can refer to the idea of disjoint union of sets (Definition 8.1). The disjoint union of two sets is a set that contains two distinct copies of each of the two sets. If an element is contained in both sets, there will be two distinct copies of it in the disjoint union.

Definition 8.1 (Disjoint union of sets)**.** The *disjoint union* (or sum) of two sets A and B is denoted $A + B$ and it is defined as

$$A + B = \{\langle 1, a\rangle \mid a \in A\} \cup \{\langle 2, b\rangle \mid b \in B\}. \tag{8.1}$$

Example 8.2. Consider the sets $\{\star, \diamond\}$ and $\{*, \dagger\}$. Their disjoint union can be represented as in Fig. 8.3.

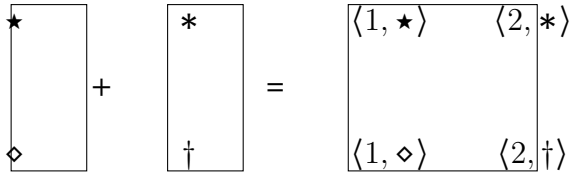


Figure 8.3: Example of a disjoint union of sets.

We can define the disjoint union of a set with itself; this corresponds to having two distinct copies of the set (Fig. 8.4).

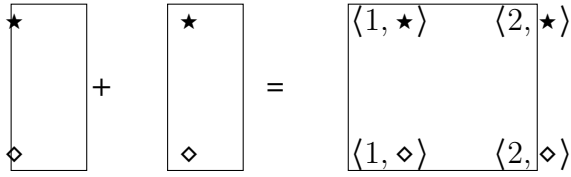
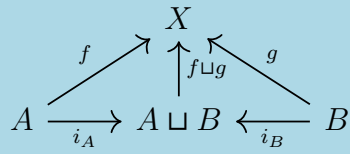


Figure 8.4: Disjoint union of a set with itself .

The disjoint union is a particular instance of the notion of "coproduct". The following definition is the general definition of coproducts for an arbitrary category.

Definition 8.3 (Coproduct)**.** Let $\mathbf{C}$ be a category and let $A, B \in \mathbf{C}$ be objects. The *coproduct* of A and B is an object $A \sqcup B \in \mathbf{C}$ together with two *inclusion morphisms* $\iota_A \colon A \to A \sqcup B$ and $\iota_B \colon B \to A \sqcup B$, such that, given any $X \in \mathbf{C}$ and morphisms $f \colon A \to X, g \colon B \to X$, there exists

a *unique* morphism $(f \sqcup g) \colon A \sqcup B \to X$ such that $f = \iota_A \fatsemi (f \sqcup g)$ and $g = \iota_B \fatsemi (f \sqcup g)$. Diagrammatically:



(8.2)

Example 8.4. The two inclusion maps for the disjoint union of sets are $\iota_1 \colon A \to A + B$ and $\iota_2 \colon B \to A + B$, defined as:

$$\begin{aligned} \iota_1(a) &= \langle 1, a \rangle \,, \\ \iota_2(b) &= \langle 2, b \rangle \,. \end{aligned} \tag{8.3}$$

Given maps $f \colon A \to X$ and $g \colon B \to X$ as in Definition 8.3, then the map $f \sqcup g$ for the disjoint union of sets is

$$\begin{aligned} f \sqcup g \colon A + B &\to X \\ y &\mapsto \begin{cases} f(y), & \text{if } y \in A, \\ g(y), & \text{if } y \in B. \end{cases} \end{aligned} \tag{8.4}$$

Note that $X \sqcup Y$ is different from $Y \sqcup X$, but the two are isomorphic (Fig. 8.5).

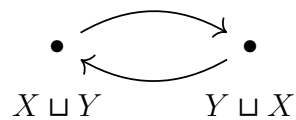


Figure 8.5: $X \sqcup Y$ and $Y \sqcup X$ are isomorphic.

For the case of motion generation, the inclusion maps are as in Fig. 8.6.

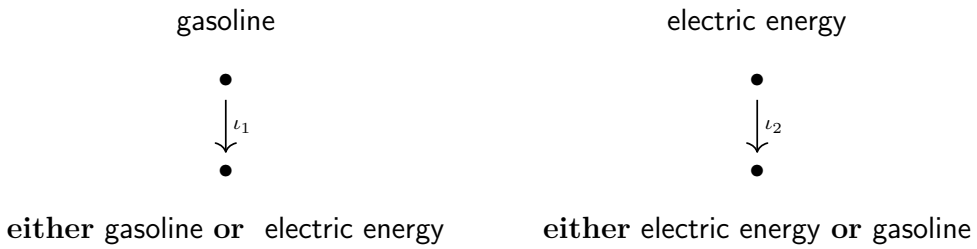


Figure 8.6: Inclusion maps for the motion generation problem.

9 Thinking about tradeoffs

10 Posets with more structure

11 Monotonicity is a fact of life

12 Functors

13 The duality of design

14 Composing without structure

15 Composing recognizing structure

16 Thinking about restrictions and alternatives

17 Parallel composition

18 Parallel composition in DP

19 Creating design problems

20 Lifting SOMETHING

21 Examples of Trace

22 Closing the loop in co-design problems

23 Ordering design problems

24 Orders on design problems and compositions

25 Relationship between products

26 Higher order design

27 Thinking about uncertainty with intervals

28 Enrichments

29 Computation