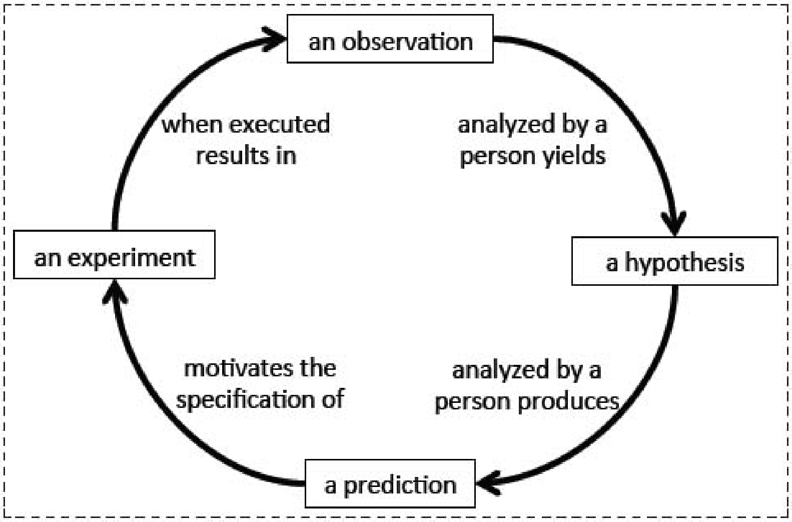
Category Theory for Scientists (Old Version)

David I. Spivak

September 17, 2013


an observation
analyzed by a person yields
a hypothesis
analyzed by a person produces
a prediction
motivates the specification of
an experiment
when executed results in

How can mathematics make this diagram meaningful?

Preface

An early version of this book was put on line in February 2013 to serve as the textbook for my course Category Theory for Scientists taught in the spring semester of 2013 at MIT. During that semester, students provided me with hundreds of comments and questions, which led to a substantial improvement (and the addition of 50 pages) to the original document.

In the summer of 2013 I signed a contract with the MIT Press to publish a new version of this work under the title *Category Theory for the Sciences*. Because I am committed to the open source development model I insisted that a version of this book, namely the one you are reading, remain freely available online. The MIT Press version will of course not be free.

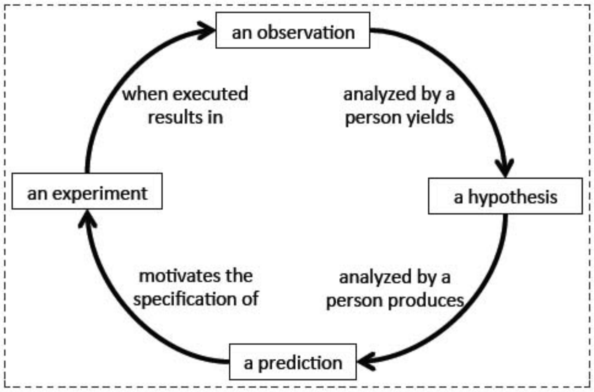
Other than the title, there are two main differences between the present version and the MIT Press version. The first difference is that I will do a full edit with the help of professional editors from the Press. The second difference is that I will write up solutions to the book's (approximately 280) exercises; some of these will be included in the published version, whereas the rest will be available by way of a password-protected page, accessible only to professors who teach the subject.

Contents

Chapter 1

Introduction

The title page of this book contains a graphic that we reproduce here.


an observation
analyzed by a person yields
a hypothesis
analyzed by a person produces
a prediction
motivates the specification of
an experiment
when executed results in

(1.1)

It is intended to evoke thoughts of the scientific method.

> A hypothesis analyzed by a person produces a prediction, which motivates the specification of an experiment, which when executed results in an observation, which analyzed by a person yields a hypothesis.

This sounds valid, and a good graphic can be exceptionally useful for leading a reader through the story that the author wishes to tell.

Interestingly, a graphic has the power to evoke feelings of understanding, without really meaning much. The same is true for text: it is possible to use a language such as English to express ideas that are never made rigorous or clear. When someone says "I believe in free will," what does she believe in? We may all have some concept of what she's saying—something we can conceptually work with and discuss or argue about. But to what extent are we all discussing the same thing, the thing she intended to convey?

Science is about agreement. When we supply a convincing argument, the result of this convincing is agreement. When, in an experiment, the observation matches the hypothesis—success!—that is agreement. When my methods make sense to you, that is agreement. When practice does not agree with theory, that is disagreement. Agreement is the good stuff in science; it's the high fives.

But it is easy to think we're in agreement, when really we're not. Modeling our thoughts on heuristics and pictures may be convenient for quick travel down the road, but we're liable to miss our turnoff at the first mile. The danger is in mistaking our convenient conceptualizations for what's actually there. It is imperative that we have the ability at any time to ground out in reality. What does that mean?

Data. Hard evidence. The physical world. It is here that science touches down and heuristics evaporate. So let's look again at the diagram on the cover. It is intended to evoke an idea of how science is performed. Is there hard evidence and data to back this theory up? Can we set up an experiment to find out whether science is actually performed according to such a protocol? To do so we have to shake off the stupor evoked by the diagram and ask the question: "what does this diagram intend to communicate?"

In this course I will use a mathematical tool called *ologs*, or ontology logs, to give some structure to the kinds of ideas that are often communicated in pictures like the one on the cover. Each olog inherently offers a framework in which to record data about the subject. More precisely it encompasses a *database schema*, which means a system of interconnected tables that are initially empty but into which data can be entered. For example consider the olog below

$$\begin{tikzcd}
\boxed{\text{a mass}} & \boxed{\begin{array}{l}\text{an object of mass}\\ m\text{ held at height }h\\ \text{above the ground}\end{array}} \arrow[l, "\text{has as mass}"'] \arrow[d, "\begin{array}{l}\text{has as height}\\ \text{in meters}\end{array}"'] \arrow[dr, "\begin{array}{l}\text{when dropped has}\\ \text{as number of sec-}\\ \text{onds till hitting the}\\ \text{ground}\end{array}"] & \\
& \boxed{\text{a real number }h} \arrow[r, "\sqrt{2h\div 9.8}"'] \arrow[ur, phantom, "?"] & \boxed{\text{a real number}}
\end{tikzcd}$$

This olog represents a framework in which to record data about objects held above the ground, their mass, their height, and a comparison (the ?-mark in the middle) between the number of seconds till they hit the ground and a certain real-valued function of their height. We will discuss ologs in detail throughout this course.

The picture in (1.1) looks like an olog, but it does not conform to the rules that we lay out for ologs in Section 2.3. In an olog, every arrow is intended to represent a mathematical function. It is difficult to imagine a function that takes in predictions and outputs experiments, but such a function is necessary in order for the arrow

$$\boxed{\text{a prediction}} \xrightarrow{\text{motivates the specification of}} \boxed{\text{an experiment}}$$

in (1.1) to make sense. To produce an experiment design from a prediction probably requires an expert, and even then the expert may be motivated to specify a different experiment on Tuesday than he is on Monday. But perhaps our criticism has led to a way forward: if we say that every arrow represents a function *when in the context of a specific expert who is actually doing the science at a specific time*, then Figure (1.1) begins to make sense. In fact, we will return to the figure in Section 5.3 (specifically Example 5.3.3.10), where background methodological context is discussed in earnest.

This course is an attempt to extol the virtues of a new branch of mathematics, called *category theory*, which was invented for powerful communication of ideas between different fields and subfields within mathematics. By powerful communication of ideas I actually mean something precise. Different branches of mathematics can be formalized into categories. These categories can then be connected together by functors. And the sense in which these functors provide powerful communication of ideas is that facts and theorems proven in one category can be transferred through a connecting functor to yield proofs of analogous theorems in another category. A functor is like a conductor of mathematical truth.

I believe that the language and toolset of category theory can be useful throughout science. We build scientific understanding by developing models, and category theory is the study of basic conceptual building blocks and how they cleanly fit together to make such models. Certain structures and conceptual frameworks show up again and again in our understanding of reality. No one would dispute that vector spaces are ubiquitous. But so are hierarchies, symmetries, actions of agents on objects, data models, global behavior emerging as the aggregate of local behavior, self-similarity, and the effect of methodological context.

Some ideas are so common that our use of them goes virtually undetected, such as set-theoretic intersections. For example, when we speak of a material that is both lightweight and ductile, we are intersecting two sets. But what is the use of even mentioning this set-theoretic fact? The answer is that when we formalize our ideas, our understanding is almost always clarified. Our ability to communicate with others is enhanced, and the possibility for developing new insights expands. And if we are ever to get to the point that we can input our ideas into computers, we will need to be able to formalize these ideas first.

It is my hope that this course will offer scientists a new vocabulary in which to think and communicate, and a new pipeline to the vast array of theorems that exist and are considered immensely powerful within mathematics. These theorems have not made their way out into the world of science, but they are directly applicable there. Hierarchies are partial orders, symmetries are group elements, data models are categories, agent actions are monoid actions, local-to-global principles are sheaves, self-similarity is modeled by operads, context can be modeled by monads.

1.1 A brief history of category theory

The paradigm shift brought on by Einstein's theory of relativity brought on the realization that there is no single perspective from which to view the world. There is no background framework that we need to find; there are infinitely many different frameworks and perspectives, and the real power lies in being able to translate between them. It is in this historical context that category theory got its start. [1]

Category theory was invented in the early 1940s by Samuel Eilenberg and Saunders Mac Lane. It was specifically designed to bridge what may appear to be two quite different fields: topology and algebra. Topology is the study of abstract shapes such as 7-dimensional spheres; algebra is the study of abstract equations such as $y^2z = x^3 - xz^2$. People had already created important and useful links (e.g. cohomology theory) between these fields, but Eilenberg and Mac Lane needed to precisely compare different links with

[1] The following history of category theory is far too brief, and perhaps reflects more of the author's aesthetic than any kind of objective truth, whatever that may mean. Here are some much better references: [Kro], [Mar1], [LM].

one another. To do so they first needed to boil down and extract the fundamental nature of these two fields. But the ideas they worked out amounted to a framework that fit not only topology and algebra, but many other mathematical disciplines as well.

At first category theory was little more than a deeply clarifying language for existing difficult mathematical ideas. However, in 1957 Alexander Grothendieck used category theory to build new mathematical machinery (new cohomology theories) that granted unprecedented insight into the behavior of algebraic equations. Since that time, categories have been built specifically to zoom in on particular features of mathematical subjects and study them with a level of acuity that is simply unavailable elsewhere.

Bill Lawvere saw category theory as a new foundation for all mathematical thought. Mathematicians had been searching for foundations in the 19th century and were reasonably satisfied with set theory as *the foundation*. But Lawvere showed that the category of sets is simply a category with certain nice properties, not necessarily the center of the mathematical universe. He explained how whole algebraic theories can be viewed as examples of a single system. He and others went on to show that higher order logic was beautifully captured in the setting of category theory (more specifically toposes). It is here also that Grothendieck and his school worked out major results in algebraic geometry.

In 1980 Joachim Lambek showed that the types and programs used in computer science form a specific kind of category. This provided a new semantics for talking about programs, allowing people to investigate how programs combine and compose to create other programs, without caring about the specifics of implementation. Eugenio Moggi brought the category theoretic notion of monads into computer science to encapsulate ideas that up to that point were considered outside the realm of such theory.

It is difficult to explain the clarity and beauty brought to category theory by people like Daniel Kan and André Joyal. They have each repeatedly extracted the essence of a whole mathematical subject to reveal and formalize a stunningly simple yet extremely powerful pattern of thinking, revolutionizing how mathematics is done.

All this time, however, category theory was consistently seen by much of the mathematical community as ridiculously abstract. But in the 21st century it has finally come to find healthy respect within the larger community of pure mathematics. It is the language of choice for graduate-level algebra and topology courses, and in my opinion will continue to establish itself as the basic framework in which mathematics is done.

As mentioned above category theory has branched out into certain areas of science as well. Baez and Dolan have shown its value in making sense of quantum physics, it is well established in computer science, and it has found proponents in several other fields as well. But to my mind, we are the very beginning of its venture into scientific methodology. Category theory was invented as a bridge and it will continue to serve in that role.

1.2 Intention of this book

The world of *applied mathematics* is much smaller than the world of *applicable mathematics*. As alluded to above, this course is intended to create a bridge between the vast array of mathematical concepts that are used daily by mathematicians to describe all manner of phenomena that arise in our studies, and the models and frameworks of scientific disciplines such as physics, computation, and neuroscience.

To the pure mathematician I'll try to prove that concepts such as categories, functors, natural transformations, limits, colimits, functor categories, sheaves, monads, and operads—concepts that are often considered too abstract for even math majors—can be communicated to scientists with no math background beyond linear algebra. If this material is as teachable as I think, it means that category theory is not esoteric but somehow well-aligned with ideas that already make sense to the scientific mind. Note, however, that this book is example-based rather than proof-based, so it may not be suitable as a reference for students of pure mathematics.

To the scientist I'll try to prove the claim that category theory includes a formal treatment of conceptual structures that the scientist sees often, perhaps without realizing that there is well-oiled mathematical machinery to be employed. We will work on the structure of information; how data is made meaningful by its connections, both internal and outreaching, to other data. Note, however, that this book should most certainly not be taken as a reference on scientific matters themselves. One should assume that any account of physics, materials science, chemistry, etc. has been oversimplified. The intention is to give a flavor of how category theory may help us model scientific ideas, not to explain these ideas in a serious way.

Data gathering is ubiquitous in science. Giant databases are currently being mined for unknown patterns, but in fact there are many (many) known patterns that simply have not been catalogued. Consider the well-known case of medical records. A patient's medical history is often known by various individual doctor-offices but quite inadequately shared between them. Sharing medical records often means faxing a hand-written note or a filled-in house-created form between offices.

Similarly, in science there exists substantial expertise making brilliant connections between concepts, but it is being conveyed in silos of English prose known as journal articles. Every scientific journal article has a methods section, but it is almost impossible to read a methods section and subsequently repeat the experiment—the English language is inadequate to precisely and concisely convey what is being done.

The first thing to understand in this course is that reusable methodologies can be formalized, and that doing so is inherently valuable. Consider the following analogy. Suppose you want to add up the area of a region in space (or the area under a curve). You break the region down into small squares, each of which you know has area A; then you count the number of squares, say n, and the result is that the region has an area of about nA. If you want a more precise and accurate result you repeat the process with half-size squares. This methodology can be used for any area-finding problem (of which there are more than a first-year calculus student generally realizes) and thus it deserves to be formalized. But once we have formalized this methodology, it can be taken to its limit and out comes integration by Riemann sums.

I intend to show that category theory is incredibly efficient as a language for experimental design patterns, introducing formality while remaining flexible. It forms a rich and tightly woven conceptual fabric that will allow the scientist to maneuver between different perspectives whenever the need arises. Once one builds that fabric for oneself, he or she has an ability to think about models in a way that simply would not occur without it. Moreover, putting ideas into the language of category theory forces a person to clarify their assumptions. This is highly valuable both for the researcher and for his or her audience.

What must be recognized in order to find value in this course is that conceptual chaos is a major problem. Creativity demands clarity of thinking, and to think clearly about a subject requires an organized understanding of how its pieces fit together. Organization and clarity also lead to better communication with others. Academics often say they are paid to think and understand, but that is not true. They are paid to think, understand,

and *communicate their findings*. Universal languages for science—languages such as calculus and differential equations, matrices, or simply graphs and pie-charts—already exist, and they grant us a cohesiveness that makes scientific research worthwhile. In this book I will attempt to show that category theory can be similarly useful in describing complex scientific understandings.

1.3 What is requested from the student

I will do my best to make clear the value of category theory in science, but I am not a scientist. To that end I am asking for your help in exploring how category theory may be useful in your specific field.

I also want you to recognize that the value of mathematics is not generally obvious at first. A good student learning a good subject with a good teacher will see something compelling almost immediately, but may not see how it will be useful in real life. This will come later. I hope you will work hard to understand even without yet knowing what its actual value in your life and research will be. Like a student of soccer is encouraged to spend hours juggling the ball when he or she could be practicing penalty shots, it is important to gain facility with the materials you will be using. Doing exercises is imperative for learning mathematics.

1.4 Category theory references

I wrote this book because the available books on category theory are almost all written for mathematicians (the rest are written for computer scientists). There is one book by Lawvere and Schanuel, called *Conceptual Mathematics* [LS], that offers category theory to a wider audience, but its style is not appropriate for this course. Still, it is very well written and clear.

The "bible" of category theory is *Categories for the working mathematician* by Mac Lane [Mac]. But as the title suggests, it was written for working mathematicians and will be quite opaque to my target audience. However, once a person has read my book, Mac Lane's book may become a valuable reference.

Other good books include Steve Awodey's book *Category theory* [Awo] and Barr and Wells book *Category theory for computing science*, [BW]. A paper by Brown and Porter called Category Theory: an abstract setting for analogy and comparison [BP1] is more in line with the style of this book, only much shorter. Online, I find wikipedia and a site called *the nlab* to be quite useful.

This book attempts to explain category theory by examples and exercises rather than by theorems and proofs. I hope this approach will be valuable to the working scientist.

1.5 Acknowledgments

I would like to express my deep appreciation for the many scientists who I have worked with over the past five years. It all started with Paea LePendu who first taught me about databases when I was naively knocking on doors in the University of Oregon computer science department. This book would never have been written if Tristan Nguyen and Dave Balaban had not noticed my work and encouraged me to continue. Dave Balaban and Peter Gates have been my scientific partners since the beginning, working hard to understand what I'm offering and working just as hard to help me understand all that I'm missing. Peter Gates has deepened my understanding of data in profound ways.

I have also been tremendously lucky to know Haynes Miller, who made it possible for me set down at MIT, with the help of Clark Barwick and Jacob Lurie. I knew that MIT would be the best place in the world for me to pursue this type of research, and it has really come through. Researchers like Markus Buehler and his graduate students Tristan Giesa and Dieter Brommer have been a pleasure to work with, and the many materials science examples scattered throughout this book is a testament to how much our work together has influenced my thinking.

I'd also like to thank my collaborators and conversation partners with whom I have discussed subjects written about in this book. Other than people mentioned above, these include Steve Awodey, Allen Brown, Adam Chlipala, Carlo Curino, Dan Dugger, Henrik Forssell, David Gepner, Jason Gross, Bob Harper, Ralph Hutchison, Robert Kent, Jack Morava, Scott Morrison, David Platt, Joey Perricone, Dylan Rupel, Guarav Singh, Sam Shames, Nat Stapleton, Patrick Schultz, Ka Yu Tam, Ryan Wisnesky, Jesse Wolfson, and Elizabeth Wood.

I would like to thank Peter Kleinhenz and Peter Gates for reading this book and providing invaluable feedback before I began teaching the 18-S996 class at MIT in Spring 2013. In particular the cover image is a mild alteration of something Gates sent me to help motivate the book to scientists. I would also like to greatly thank the 18-S996 course grader Darij Grinberg, who was not only the best grader I've had in my 14 years of teaching, but gave me more comments than anyone else on the book itself. I'd also like to thank the students from the 18-S996 class at MIT who helped me find typos, pointed me to unclear explanations, and generally helped me improve the book in many ways. Other than the people listed above, these include Aaron Brookner, Leon Dimas, Dylan Erb, Deokhwan Kim, Taesoo Kim, Owen Lewis, Yair Shenfeld, and Adam Strandberg.

I would like to thank my teacher, Peter Ralston, who taught me to repeatedly question the obvious. My ability to commit to a project like this one and to see it to fruition has certainly been enhanced since studying with him.

Finally, I acknowledge my appreciation for support from the Office of Naval Research [2] without which this book would not have been remotely possible. I believe that their funding of basic research is an excellent way of ensuring that the US remains a global leader in the years to come.

[2] Grant numbers: N000140910466, N000141010841, N000141310260

Chapter 2

The category of sets

The theory of sets was invented as a foundation for all of mathematics. The notion of sets and functions serves as a basis on which to build our intuition about categories in general. In this chapter we will give examples of sets and functions and then move on to discuss commutative diagrams. At this point we can introduce ologs which will allow us to use the language of category theory to speak about real world concepts. Then we will introduce limits and colimits, and their universal properties. All of this material is basic set theory, but it can also be taken as an investigation of our first category, the *category of sets*, which we call **Set**. We will end this chapter with some other interesting constructions in **Set** that do not fit into the previous sections.

2.1 Sets and functions

2.1.1 Sets

In this course I'll assume you know what a set is. We can think of a set X as a collection of things $x \in X$, each of which is recognizable as being in X and such that for each pair of named elements $x, x' \in X$ we can tell if $x = x'$ or not. [1] The set of pendulums is the collection of things we agree to call pendulums, each of which is recognizable as being a pendulum, and for any two people pointing at pendulums we can tell if they're pointing at the same pendulum or not.

Notation 2.1.1.1. The symbol $\emptyset$ denotes the set with no elements. The symbol $\mathbb{N}$ denotes the set of natural numbers, which we can write as

$$\mathbb{N} := \{0, 1, 2, 3, 4, \ldots, 877, \ldots\}.$$

The symbol $\mathbb{Z}$ denotes the set of integers, which contains both the natural numbers and their negatives,

$$\mathbb{Z} := \{\ldots, -551, \ldots, -2, -1, 0, 1, 2, \ldots\}.$$

If A and B are sets, we say that A is a *subset* of B, and write $A \subseteq B$, if every element of A is an element of B. So we have $\mathbb{N} \subseteq \mathbb{Z}$. Checking the definition, one sees that

[1] Note that the symbol x', read "x-prime", has nothing to do with calculus or derivatives. It is simply notation that we use to name a symbol that is suggested as being somehow like x. This suggestion of kinship between x and x' is meant only as an aid for human cognition, and not as part of the mathematics.

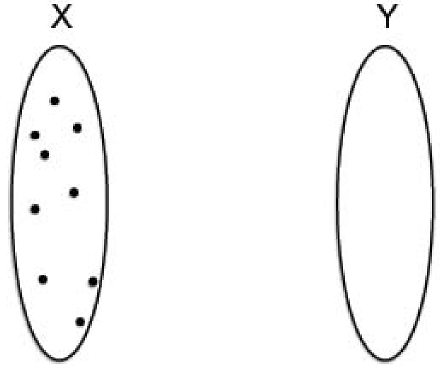


Figure 2.1: A set X with 9 elements and a set Y with no elements, $Y = \varnothing$.

for any set A, we have (perhaps uninteresting) subsets $\varnothing \subseteq A$ and $A \subseteq A$. We can use *set-builder notation* to denote subsets. For example the set of even integers can be written $\{n \in \mathbb{Z} \mid n \text{ is even}\}$. The set of integers greater than 2 can be written in many ways, such as

$$\{n \in \mathbb{Z} \mid n > 2\} \qquad \text{or} \qquad \{n \in \mathbb{N} \mid n > 2\} \qquad \text{or} \qquad \{n \in \mathbb{N} \mid n \geqslant 3\}.$$

The symbol $\exists$ means "there exists". So we could write the set of even integers as

$$\{n \in \mathbb{Z} \mid n \text{ is even}\} \qquad = \qquad \{n \in \mathbb{Z} \mid \exists m \in \mathbb{Z} \text{ such that } 2m = n\}.$$

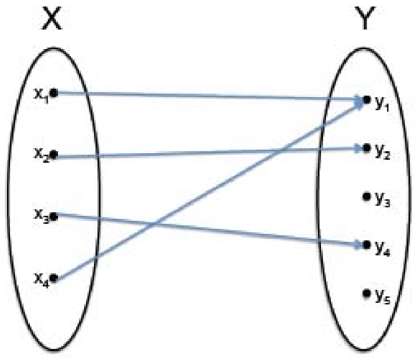
The symbol $\exists!$ means "there exists a unique". So the statement "$\exists! x \in \mathbb{R}$ such that $x^2 = 0$" means that there is one and only one number whose square is 0. Finally, the symbol $\forall$ means "for all". So the statement "$\forall m \in \mathbb{N}\ \exists n \in \mathbb{N}$ such that $m < n$" means that for every number there is a bigger one.

As you may have noticed, we use the colon-equals notation " $A := XYZ$ " to mean something like "define A to be XYZ". That is, a colon-equals declaration is not denoting a fact of nature (like $2 + 2 = 4$), but a choice of the speaker. It just so happens that the notation above, such as $\mathbb{N} := \{0, 1, 2, \ldots\}$, is a widely-held choice.

Exercise 2.1.1.2. Let $A = \{1, 2, 3\}$. What are all the subsets of A? Hint: there are 8. ◊

2.1.2 Functions

If X and Y are sets, then a *function f from X to Y*, denoted $f\colon X \to Y$, is a mapping that sends each element $x \in X$ to an element of Y, denoted $f(x) \in Y$. We call X the *domain* of the function f and we call Y the *codomain* of f.



(2.2)

Note that for every element $x \in X$, there is exactly one arrow emanating from x, but for an element $y \in Y$, there can be several arrows pointing to y, or there can be no arrows pointing to y.

Application 2.1.2.1. In studying the mechanics of materials, one wishes to know how a material responds to tension. For example a rubber band responds to tension differently than a spring does. To each material we can associate a force-extension curve, recording how much force the material carries when extended to various lengths. Once we fix a methodology for performing experiments, finding a material's force-extension curve would ideally constitute a function from the set of materials to the set of curves. [2]

◊◊

Exercise 2.1.2.2. Here is a simplified account of how the brain receives light. The eye contains about 100 million photoreceptor (PR) cells. Each connects to a retinal ganglion (RG) cell. No PR cell connects to two different RG cells, but usually many PR cells can attach to a single RG cell.

Let PR denote the set of photoreceptor cells and let RG denote the set of retinal ganglion cells.

a.) According to the above account, does the connection pattern constitute a function $RG \to PR$, a function $PR \to RG$ or neither one?

b.) Would you guess that the connection pattern that exists between other areas of the brain are "function-like"?

◊

Example 2.1.2.3. Suppose that X is a set and $X' \subseteq X$ is a subset. Then we can consider the function $X' \to X$ given by sending every element of X' to "itself" as an element of X. For example if $X = \{a, b, c, d, e, f\}$ and $X' = \{b, d, e\}$ then $X' \subseteq X$ and we turn that into the function $X' \to X$ given by $b \mapsto b, d \mapsto d, e \mapsto e$. [3]

As a matter of notation, we may sometimes say something like the following: Let X be a set and let $i \colon X' \subseteq X$ be a subset. Here we are making clear that X' is a subset of X, but that i is the name of the associated function.

[2] In reality, different samples of the same material, say samples of different sizes or at different temperatures, may have different force-extension curves. If we want to see this as a true function whose codomain is curves it should have as domain something like the set of material samples.

[3] This kind of arrow, $\mapsto$, is read aloud as "maps to". A function $f \colon X \to Y$ means a rule for assigning to each element $x \in X$ an element $f(x) \in Y$. We say that "x maps to $f(x)$" and write $x \mapsto f(x)$.

Exercise 2.1.2.4. Let $f\colon \mathbb{N} \to \mathbb{N}$ be the function that sends every natural number to its square, e.g. $f(6) = 36$. First fill in the blanks below, then answer a question.

a.) $2 \mapsto$ ________

b.) $0 \mapsto$ ________

c.) $-2 \mapsto$ ________

d.) $5 \mapsto$ ________

e.) Consider the symbol → and the symbol ↦. What is the difference between how these two symbols are used in this book?

◊

Given a function $f\colon X \to Y$, the elements of Y that have at least one arrow pointing to them are said to be *in the image* of f; that is we have

$$\mathrm{im}(f) := \{y \in Y \mid \exists x \in X \text{ such that } f(x) = y\}. \tag{2.3}$$

Exercise 2.1.2.5. If $f\colon X \to Y$ is depicted by (2.2) above, write its image, $\mathrm{im}(f)$ as a set. ◊

Given a function $f\colon X \to Y$ and a function $g\colon Y \to Z$, where the codomain of f is the same set as the domain of g (namely Y), we say that f and g are composable

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

The *composition of f and g* is denoted by $g \circ f\colon X \to Z$.

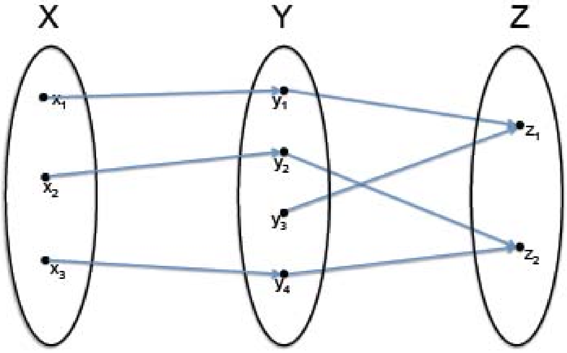


Figure 2.4: Functions $f\colon X \to Y$ and $g\colon Y \to Z$ compose to a function $g \circ f\colon X \to Z$; just follow the arrows.

Let X and Y be sets. We write $\mathrm{Hom}_{\mathbf{Set}}(X, Y)$ to denote the set of functions $X \to Y$. [4] Note that two functions $f, g\colon X \to Y$ are equal if and only if for every element $x \in X$ we have $f(x) = g(x)$.

Exercise 2.1.2.6. Let $A = \{1, 2, 3, 4, 5\}$ and $B = \{x, y\}$.

[4] The strange notation $\mathrm{Hom}_{\mathbf{Set}}(-,-)$ will make more sense later, when it is seen as part of a bigger story.

a.) How many elements does $\text{Hom}_{\mathbf{Set}}(A, B)$ have?

b.) How many elements does $\text{Hom}_{\mathbf{Set}}(B, A)$ have?

◊

Exercise 2.1.2.7.

a.) Find a set A such that for all sets X there is exactly one element in $\text{Hom}_{\mathbf{Set}}(X, A)$. Hint: draw a picture of proposed A's and X's.

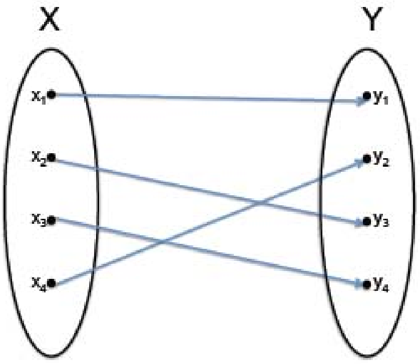
b.) Find a set B such that for all sets X there is exactly one element in $\text{Hom}_{\mathbf{Set}}(B, X)$.

◊

For any set X, we define the *identity function on* X, denoted $\text{id}_X : X \to X$, to be the function such that for all $x \in X$ we have $\text{id}_X(x) = x$.

Definition 2.1.2.8 (Isomorphism)**.** Let X and Y be sets. A function $f : X \to Y$ is called an *isomorphism*, denoted $f : X \xrightarrow{\cong} Y$, if there exists a function $g : Y \to X$ such that $g \circ f = \text{id}_X$ and $f \circ g = \text{id}_Y$. We also say that f is *invertible* and we say that g is *the inverse* of f. If there exists an isomorphism $X \xrightarrow{\cong} Y$ we say that X and Y are *isomorphic* sets and may write $X \cong Y$.

Example 2.1.2.9. If X and Y are sets and $f : X \to Y$ is an isomorphism then the analogue of Diagram 2.2 will look like a perfect matching, more often called a *one-to-one correspondence*. That means that no two arrows will hit the same element of Y, and every element of Y will be in the image. For example, the following depicts an isomorphism $X \xrightarrow{\cong} Y$.



Application 2.1.2.10. There is an isomorphism between the set Nuc_{DNA} of nucleotides found in DNA and the set Nuc_{RNA} of nucleotides found in RNA. Indeed both sets have four elements, so there are 24 different isomorphisms. But only one is useful. Before we say which one it is, let us say there is also an isomorphism $\text{Nuc}_{\text{DNA}} \cong \{A, C, G, T\}$ and an isomorphism $\text{Nuc}_{\text{RNA}} \cong \{A, C, G, U\}$, and we will use the letters as abbreviations for the nucleotides.

The convenient isomorphism $\text{Nuc}_{\text{DNA}} \xrightarrow{\cong} \text{Nuc}_{\text{RNA}}$ is that given by RNA transcription; it sends

$$A \mapsto U, C \mapsto G, G \mapsto C, T \mapsto A.$$

(See also Application 4.1.2.19.) There is also an isomorphism $\mathrm{Nuc}_{\mathrm{DNA}} \xrightarrow{\cong} \mathrm{Nuc}_{\mathrm{DNA}}$ (the matching in the double-helix) given by

$$A \mapsto T, C \mapsto G, G \mapsto C, T \mapsto A.$$

Protein production can be modeled as a function from the set of 3-nucleotide sequences to the set of eukaryotic amino acids. However, it cannot be an isomorphism because there are $4^3 = 64$ triplets of RNA nucleotides, but only 21 eukaryotic amino acids.

◊◊

Exercise 2.1.2.11. Let $n \in \mathbb{N}$ be a natural number and let X be a set with exactly n elements.

a.) How many isomorphisms are there from X to itself?

b.) Does your formula from part a.) hold when $n = 0$?

◊

Lemma 2.1.2.12. *The following facts hold about isomorphism.*

1. *Any set A is isomorphic to itself; i.e. there exists an isomorphism $A \xrightarrow{\cong} A$.*

2. *For any sets A and B, if A is isomorphic to B then B is isomorphic to A.*

3. *For any sets A, B, and C, if A is isomorphic to B and B is isomorphic to C then A is isomorphic to C.*

Proof. 1. The identity function $\mathrm{id}_A \colon A \to A$ is invertible; its inverse is id_A because $\mathrm{id}_A \circ \mathrm{id}_A = \mathrm{id}_A$.

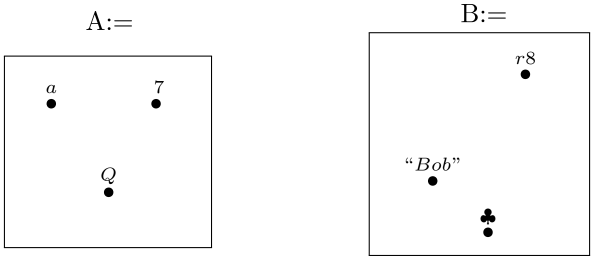
2. If $f \colon A \to B$ is invertible with inverse $g \colon B \to A$ then g is an isomorphism with inverse f.

3. If $f \colon A \to B$ and $f' \colon B \to C$ are each invertible with inverses $g \colon B \to A$ and $g' \colon C \to B$ then the following calculations show that $f' \circ f$ is invertible with inverse $g \circ g'$:

$$(f' \circ f) \circ (g \circ g') = f' \circ (f \circ g) \circ g' = f' \circ \mathrm{id}_B \circ g' = f' \circ g' = \mathrm{id}_C$$
$$(g \circ g') \circ (f' \circ f) = g \circ (g' \circ f') \circ f = g \circ \mathrm{id}_B \circ f = g \circ f = \mathrm{id}_A$$

□

Exercise 2.1.2.13. Let A and B be the sets drawn below:



Note that the sets A and B are isomorphic. Supposing that $f\colon B \to \{1,2,3,4,5\}$ sends "Bob" to 1, sends ♣ to 3, and sends $r8$ to 4, is there a canonical function $A \to \{1,2,3,4,5\}$ corresponding to f? [5] ◊

Exercise 2.1.2.14. Find a set A such that for any set X there is a isomorphism of sets

$$X \cong \mathrm{Hom}_{\mathbf{Set}}(A, X).$$

Hint: draw a picture of proposed A's and X's. ◊

For any natural number $n \in \mathbb{N}$, define a set

$$\underline{n} := \{1, 2, 3, \ldots, n\}. \tag{2.6}$$

So, in particular, $\underline{2} = \{1,2\}, \underline{1} = \{1\}$, and $\underline{0} = \varnothing$.

Let A be any set. A function $f\colon \underline{n} \to A$ can be written as a sequence

$$f = (f(1), f(2), \ldots, f(n)).$$

Exercise 2.1.2.15.

a.) Let $A = \{a,b,c,d\}$. If $f\colon \underline{10} \to A$ is given by (a,b,c,c,b,a,d,d,a,b), what is $f(4)$?

b.) Let $s\colon \underline{7} \to \mathbb{N}$ be given by $s(i) = i^2$. Write s out as a sequence.

◊

Definition 2.1.2.16. Cardinality of finite sets][

Let A be a set and $n \in \mathbb{N}$ a natural number. We say that A is *has cardinality* n, denoted

$$|A| = n,$$

if there exists an isomorphism of sets $A \cong \underline{n}$. If there exists some $n \in \mathbb{N}$ such that A has cardinality n then we say that A is *finite*. Otherwise, we say that A is *infinite* and write $|A| \geqslant \infty$.

Exercise 2.1.2.17.

a.) Let $A = \{5,6,7\}$. What is $|A|$?

b.) What is $|\mathbb{N}|$?

c.) What is $|\{n \in \mathbb{N} \mid n \leqslant 5\}|$?

◊

Lemma 2.1.2.18. *Let A and B be finite sets. If there is an isomorphism of sets $f\colon A \to B$ then the two sets have the same cardinality, $|A| = |B|$.*

Proof. Suppose $f\colon A \to B$ is an isomorphism. If there exists natural numbers $m, n \in \mathbb{N}$ and isomorphisms $a\colon \underline{m} \xrightarrow{\cong} A$ and $b\colon \underline{n} \xrightarrow{\cong} B$ then $\underline{m} \xrightarrow{a^{-1}} A \xrightarrow{f} B \xrightarrow{b} \underline{n}$ is an isomorphism. One can prove by induction that the sets $\underline{m}$ and $\underline{n}$ are isomorphic if and only if $m = n$. □

[5] Canonical means something like "best choice", a choice that stands out as the only reasonable one.

2.2 Commutative diagrams

At this point it is difficult to precisely define diagrams or commutative diagrams in general, but we can give the heuristic idea. [6] Consider the following picture:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "h"'] & B \arrow[d, "g"] \\ & C \end{tikzcd} \tag{2.7}$$

We say this is a *diagram of sets* if each of A, B, C is a set and each of f, g, h is a function. We say this diagram *commutes* if $g \circ f = h$. In this case we refer to it as a commutative triangle of sets.

Application 2.2.1.1. The central dogma of molecular biology is that "DNA codes for RNA codes for protein". That is, there is a function from DNA triplets to RNA triplets and a function from RNA triplets to amino acids. But sometimes we just want to discuss the translation from DNA to amino acids, and this is the composite of the other two. The commutative diagram is a picture of this fact.

◊◊

Consider the following picture:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\ C \arrow[r, "i"'] & D \end{tikzcd}$$

We say this is a *diagram of sets* if each of A, B, C, D is a set and each of f, g, h, i is a function. We say this diagram *commutes* if $g \circ f = i \circ h$. In this case we refer to it as a commutative square of sets.

Application 2.2.1.2. Given a physical system S, there may be two mathematical approaches $f \colon S \to A$ and $g \colon S \to B$ that can be applied to it. Either of those results in a prediction of the same sort, $f' \colon A \to P$ and $g' \colon B \to P$. For example, in mechanics we can use either Lagrangian approach or the Hamiltonian approach to predict future states. To say that the diagram

$$\begin{tikzcd} S \arrow[r] \arrow[d] & A \arrow[d] \\ B \arrow[r] & P \end{tikzcd}$$

commutes would say that these approaches give the same result.

◊◊

And so on. Note that diagram (2.7) is considered to be the same diagram as each of

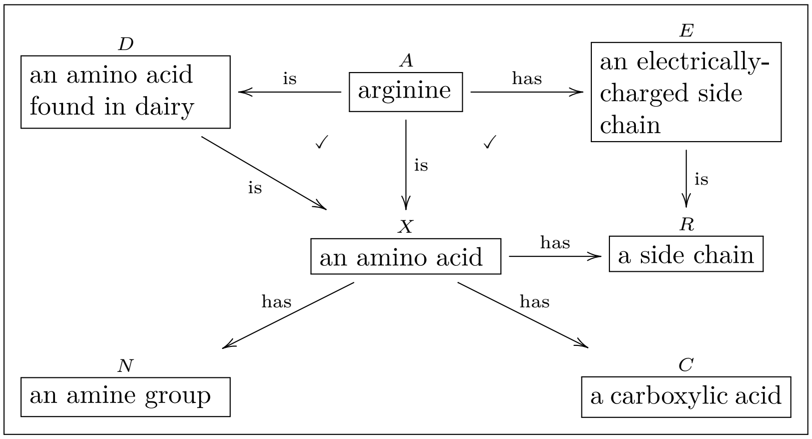
[6] We will define commutative diagrams precisely in Section 4.5.2.

the following:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[dl, "g"] \\ C & \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "f"] \arrow[rr, bend right, "h"'] & B \arrow[r, "g"] & C \end{tikzcd} \qquad \begin{tikzcd} B \arrow[dr, "g"] & \\ & C \\ A \arrow[uu, "f"] \arrow[ur, "h"'] & \end{tikzcd}$$

2.3 Ologs

In this course we will ground the mathematical ideas in applications whenever possible. To that end we introduce ologs, which will serve as a bridge between mathematics and various conceptual landscapes. The following material is taken from [SK], an introduction to ologs.



(2.8)

2.3.1 Types

A type is an abstract concept, a distinction the author has made. We represent each type as a box containing a *singular indefinite noun phrase.* Each of the following four boxes is a type:

$$\boxed{\text{a man}} \qquad \boxed{\text{an automobile}} \tag{2.9}$$

$$\boxed{\text{a pair } (a, w)\text{, where } w \text{ is a woman and } a \text{ is an automobile}} \qquad \boxed{\text{a pair } (a, w) \text{ where } w \text{ is a woman and } a \text{ is a blue automobile owned by } w}$$

Each of the four boxes in (2.9) represents a type of thing, a whole class of things, and the label on that box is what one should call *each example* of that class. Thus ⌜a man⌝ does not represent a single man, but the set of men, each example of which is called “a man”. Similarly, the bottom right box represents an abstract type of thing, which probably has more than a million examples, but the label on the box indicates the common name for each such example.

Typographical problems emerge when writing a text-box in a line of text, e.g. the text-box $\boxed{\text{a man}}$ seems out of place here, and the more in-line text-boxes there are, the worse it gets. To remedy this, I will denote types which occur in a line of text with corner-symbols; e.g. I will write ⌜a man⌝ instead of $\boxed{\text{a man}}$.

2.3.1.1 Types with compound structures

Many types have compound structures; i.e. they are composed of smaller units. Examples include

$$\boxed{\begin{array}{l}\text{a man and}\\ \text{a woman}\end{array}} \qquad \boxed{\begin{array}{l}\text{a food portion } f \text{ and}\\ \text{a child } c \text{ such that } c\\ \text{ate all of } f\end{array}} \qquad \boxed{\begin{array}{l}\text{a triple } (p,a,j) \text{ where } p \text{ is}\\ \text{a paper, } a \text{ is an author of}\\ p\text{, and } j \text{ is a journal in}\\ \text{which } p \text{ was published}\end{array}} \tag{2.10}$$

It is good practice to declare the variables in a "compound type", as I did in the last two cases of (2.10). In other words, it is preferable to replace the first box above with something like

$$\boxed{\begin{array}{l}\text{a man } m \text{ and}\\ \text{a woman } w\end{array}} \qquad \text{or} \qquad \boxed{\begin{array}{l}\text{a pair } (m,w)\\ \text{where } m \text{ is a man}\\ \text{and } w \text{ is a woman}\end{array}}$$

so that the variables (m,w) are clear.

Rules of good practice 2.3.1.2. A type is presented as a text box. The text in that box should

(i) begin with the word "a" or "an";

(ii) refer to a distinction made and recognizable by the olog's author;

(iii) refer to a distinction for which instances can be documented;

(iv) declare all variables in a compound structure.

The first, second, and third rules ensure that the class of things represented by each box appears to the author as a well-defined set. The fourth rule encourages good "readability" of arrows, as will be discussed next in Section 2.3.2.

I will not always follow the rules of good practice throughout this document. I think of these rules being followed "in the background" but that I have "nicknamed" various boxes. So ⌜Steve⌝ may stand as a nickname for ⌜a thing classified as Steve⌝ and ⌜arginine⌝ as a nickname for ⌜a molecule of arginine⌝. However, when pressed, one should always be able to rename each type according to the rules of good practice.

2.3.2 Aspects

An aspect of a thing x is a way of viewing it, a particular way in which x can be regarded or measured. For example, a woman can be regarded as a person; hence "being a person" is an aspect of a woman. A molecule has a molecular mass (say in daltons), so "having a molecular mass" is an aspect of a molecule. In other words, by *aspect* we simply mean a function. The domain A of the function $f\colon A \to B$ is the thing we are measuring, and the codomain is the set of possible "answers" or results of the measurement.

$$\begin{tikzcd}\boxed{\text{a woman}} \arrow[r, "\text{is}"] & \boxed{\text{a person}}\end{tikzcd} \tag{2.11}$$

$$\begin{tikzcd}\boxed{\text{a molecule}} \arrow[r, "\text{has as molecular mass (Da)}"] & \boxed{\text{a positive real number}}\end{tikzcd} \tag{2.12}$$

So for the arrow in (2.11), the domain is the set of women (a set with perhaps 3 billion elements); the codomain is the set of persons (a set with perhaps 6 billion elements). We can imagine drawing an arrow from each dot in the "woman" set to a unique dot in the "person" set, just as in (2.2). No woman points to two different people, nor to zero people — each woman is exactly one person — so the rules for a function are satisfied. Let us now concentrate briefly on the arrow in (2.12). The domain is the set of molecules, the codomain is the set $\mathbb{R}_{>0}$ of positive real numbers. We can imagine drawing an arrow from each dot in the "molecule" set to a single dot in the "positive real number" set. No molecule points to two different masses, nor can a molecule have no mass: each molecule has exactly one mass. Note however that two different molecules can point to the same mass.

2.3.2.1 Invalid aspects

I tried above to clarify what it is that makes an aspect "valid", namely that it must be a "functional relationship." In this subsection I will show two arrows which on their face may appear to be aspects, but which on closer inspection are not functional (and hence are not valid as aspects).

Consider the following two arrows:

$$\begin{tikzcd}\boxed{\text{a person}} \arrow[r, "\text{has}"] & \boxed{\text{a child}}\end{tikzcd} \tag{2.13*}$$

$$\begin{tikzcd}\boxed{\text{a mechanical pencil}} \arrow[r, "\text{uses}"] & \boxed{\text{a piece of lead}}\end{tikzcd} \tag{2.14*}$$

A person may have no children or may have more than one child, so the first arrow is invalid: it is not a function. Similarly, if we drew an arrow from each mechanical pencil to each piece of lead it uses, it would not be a function.

Warning 2.3.2.2. The author of an olog has a world-view, some fragment of which is captured in the olog. When person A examines the olog of person B, person A may or may not "agree with it." For example, person B may have the following olog

$$\boxed{\begin{tikzcd} & \boxed{\text{a marriage}} \arrow[dl, "\text{includes}"'] \arrow[dr, "\text{includes}"] & \\ \boxed{\text{a man}} & & \boxed{\text{a woman}}\end{tikzcd}}$$

which associates to each marriage a man and a woman. Person A may take the position that some marriages involve two men or two women, and thus see B's olog as "wrong."

Such disputes are not "problems" with either A's olog or B's olog, they are discrepancies between world-views. Hence, throughout this paper, a reader R may see a displayed olog and notice a discrepancy between R's world-view and my own, but R should not worry that this is a problem. This is not to say that ologs need not follow rules, but instead that the rules are enforced to ensure that an olog is structurally sound, rather than that it "correctly reflects reality," whatever that may mean.

Consider the aspect ⌜an object⌝ $\xrightarrow{\text{has}}$ ⌜a weight⌝. At some point in history, this would have been considered a valid function. Now we know that the same object would have a different weight on the moon than it has on earth. Thus as world-views change, we often need to add more information to our olog. Even the validity of ⌜an object on earth⌝ $\xrightarrow{\text{has}}$ ⌜a weight⌝ is questionable. However to build a model we need to choose a level of granularity and try to stay within it, or the whole model evaporates into the nothingness of truth!

Remark 2.3.2.3. In keeping with Warning 2.3.2.2, the arrows (2.13*) and (2.14*) may not be wrong but simply reflect that the author has a strange world-view or a strange vocabulary. Maybe the author believes that every mechanical pencil uses exactly one piece of lead. If this is so, then ⌜a mechanical pencil⌝ $\xrightarrow{\text{uses}}$ ⌜a piece of lead⌝ is indeed a valid aspect! Similarly, suppose the author meant to say that each person *was once* a child, or that a person has an inner child. Since every person has one and only one inner child (according to the author), the map ⌜a person⌝ $\xrightarrow{\text{has as inner child}}$ ⌜a child⌝ is a valid aspect. We cannot fault the olog if the author has a view, but note that we have changed the name of the label to make his or her intention more explicit.

2.3.2.4 Reading aspects and paths as English phrases

Each arrow (aspect) $X \xrightarrow{f} Y$ can be read by first reading the label on its source box (domain of definition) X, then the label on the arrow f, and finally the label on its target box (set of values) Y. For example, the arrow

$$\boxed{\begin{tikzcd} \boxed{\text{a book}} \arrow[r, "\text{has as first author}"] & \boxed{\text{a person}} \end{tikzcd}} \tag{2.15}$$

is read "a book has as first author a person".

Remark 2.3.2.5. Note that the map in (2.15) is a valid aspect, but that a similarly benign-looking map ⌜a book⌝ $\xrightarrow{\text{has as author}}$ ⌜a person⌝ would not be valid, because it is not functional. The authors of an olog must be vigilant about this type of mistake because it is easy to miss and it can corrupt the olog.

Sometimes the label on an arrow can be shortened or dropped altogether if it is obvious from context. We will discuss this more in Section 2.3.3 but here is a common example from the way I write ologs.

$$\begin{tikzcd} & \overset{A}{\boxed{\begin{array}{l}\text{a pair } (x,y) \text{ where}\\ x \text{ and } y \text{ are integers}\end{array}}} \arrow[dl, "x"'] \arrow[dr, "y"] & \\ \overset{B}{\boxed{\text{an integer}}} & & \overset{B}{\boxed{\text{an integer}}} \end{tikzcd} \tag{2.16}$$

Neither arrow is readable by the protocol given above (e.g. "a pair (x, y) where x and y are integers x an integer" is not an English sentence), and yet it is obvious what each map means. For example, given $(8, 11)$ in A, arrow x would yield 8 and arrow y would yield 11. The label x can be thought of as a nickname for the full name "yields, via the value of x," and similarly for y. I do not generally use the full name for fear that the olog would become cluttered with text.

One can also read paths through an olog by inserting the word "which" after each intermediate box. [7] For example the following olog has two paths of length 3 (counting arrows in a chain):

$$\begin{tikzcd} \boxed{\text{a child}} \arrow[r, "\text{is}"] & \boxed{\text{a person}} \arrow[rr, "\text{has as parents}"] \arrow[dr, "\text{has, as birthday}"'] & & \boxed{\begin{array}{l}\text{a pair } (w,m)\\ \text{where } w \text{ is a}\\ \text{woman and } m\\ \text{is a man}\end{array}} \arrow[r, "w"] & \boxed{\text{a woman}} \\ & & \boxed{\text{a date}} \arrow[r, "\text{includes}"] & \boxed{\text{a year}} & \end{tikzcd} \tag{2.17}$$

The top path is read "a child is a person, who has as parents a pair (w, m) where w is a woman and m is a man, which yields, via the value of w, a woman." The reader should read and understand the content of the bottom path, which associates to every child a year.

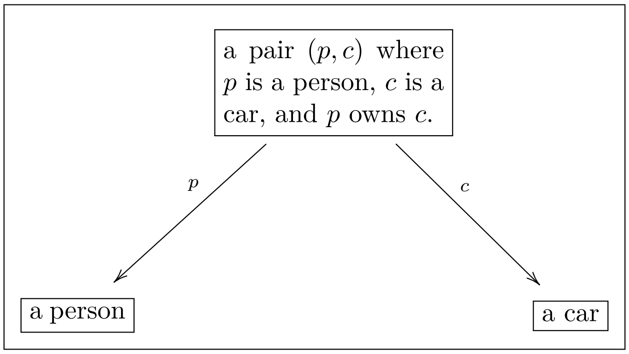
2.3.2.6 Converting non-functional relationships to aspects

There are many relationships that are not functional, and these cannot be considered aspects. Often the word "has" indicates a relationship — sometimes it is functional as in ⌜a person⌝ $\xrightarrow{\text{has}}$ ⌜a stomach⌝, and sometimes it is not, as in ⌜a father⌝ $\xrightarrow{\text{has}}$ ⌜a child⌝. Obviously, a father may have more than one child. This one is easily fixed by realizing that the arrow should go the other way: there is a function ⌜a child⌝ $\xrightarrow{\text{has}}$ ⌜a father⌝.

What about ⌜a person⌝ $\xrightarrow{\text{owns}}$ ⌜a car⌝. Again, a person may own no cars or more than one car, but this time a car can be owned by more than one person too. A quick fix would be to replace it by ⌜a person⌝ $\xrightarrow{\text{owns}}$ ⌜a set of cars⌝. This is ok, but the relationship between ⌜a car⌝ and ⌜a set of cars⌝ then becomes an issue to deal with later. There is

[7] If the intended elements of an intermediate box are humans, it is polite to use "who" rather than "which", and other such conventions may be upheld if one so desires.

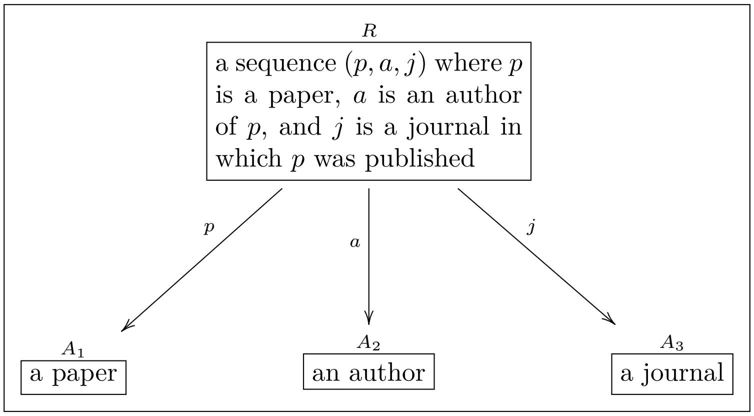
another way to indicate such “non-functional” relationships. In this case it would look like this:


a pair (p, c) where p is a person, c is a car, and p owns c.
p
c
a person
a car

This setup will ensure that everything is properly organized. In general, relationships can involve more than two types, and the general situation looks like this

$$\begin{tikzcd}
 & & \boxed{R} \arrow[dll] \arrow[dl] \arrow[dr] & \\
\boxed{A_1} & \boxed{A_2} & \cdots & \boxed{A_n}
\end{tikzcd}$$

For example,


R
a sequence (p, a, j) where p is a paper, a is an author of p, and j is a journal in which p was published
p
a
j
A_1
a paper
A_2
an author
A_3
a journal

Exercise 2.3.2.7. On page 25 we indicate a so-called invalid aspect, namely

$$\boxed{\text{a person}} \xrightarrow{\text{has}} \boxed{\text{a child}} \qquad (2.13*)$$

Create a (valid) olog that captures the parent-child relationship; your olog should still have boxes ⌜a person⌝ and ⌜a child⌝ but may have an additional box. ◊

Rules of good practice 2.3.2.8. An aspect is presented as a labeled arrow, pointing from a source box to a target box. The arrow text should

(i) begin with a verb;

(ii) yield an English sentence, when the source-box text followed by the arrow text followed by the target-box text is read; and

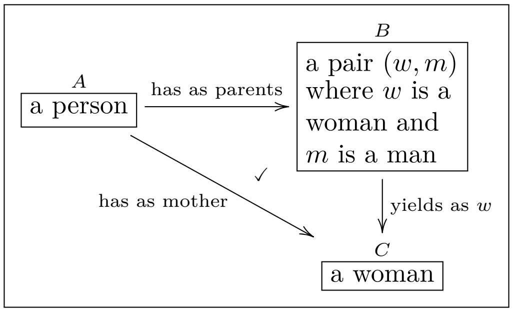
(iii) refer to a functional relationship: each instance of the source type should give rise to a specific instance of the target type.

2.3.3 Facts

In this section I will discuss facts, which are simply "path equivalences" in an olog. It is the notion of path equivalences that make category theory so powerful.

A *path* in an olog is a head-to-tail sequence of arrows. That is, any path starts at some box B_0, then follows an arrow emanating from B_0 (moving in the appropriate direction), at which point it lands at another box B_1, then follows any arrow emanating from B_1, etc, eventually landing at a box B_n and stopping there. The number of arrows is the *length* of the path. So a path of length 1 is just an arrow, and a path of length 0 is just a box. We call B_0 the *source* and B_n the *target* of the path.

Given an olog, the author may want to declare that two paths are equivalent. For example consider the two paths from A to C in the olog



(2.18)

We know as English speakers that a woman parent is called a mother, so these two paths $A \to C$ should be equivalent. A more mathematical way to say this is that the triangle in Olog (2.18) *commutes*. That is, path equivalences are simply commutative diagrams as in Section 2.2. In the example above we concisely say "a woman parent is equivalent to a mother." We declare this by defining the diagonal map in (2.18) to be *the composition* of the horizontal map and the vertical map.

I generally prefer to indicate a commutative diagram by drawing a check-mark, ✓, in the region bounded by the two paths, as in Olog (2.18). Sometimes, however, one cannot do this unambiguously on the 2-dimensional page. In such a case I will indicate the commutative diagrams (fact) by writing an equation. For example to say that the diagram

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[d, "g"] \\
C \arrow[r, "i"'] & D
\end{tikzcd}$$

commutes, we could either draw a checkmark inside the square or write the equation

$A\ f\ g \simeq A\ h\ i$ above it. [8] Either way, it means that "f then g" is equivalent to "h then i".

Here is another, more scientific example:

$$\begin{tikzcd}
\boxed{\text{a DNA sequence}} \arrow[r, "\text{is transcribed to}"] \arrow[dr, "\text{codes for}"'] \arrow[r, phantom, shift right=6, "\checkmark"] & \boxed{\text{an RNA sequence}} \arrow[d, "\text{is translated to}"] \\
 & \boxed{\text{a protein}}
\end{tikzcd}$$

Note how this diagram gives us the established terminology for the various ways in which DNA, RNA, and protein are related in this context.

Exercise 2.3.3.1. Create an olog for human nuclear biological families that includes the concept of person, man, woman, parent, father, mother, and child. Make sure to label all the arrows, and make sure each arrow indicates a valid aspect in the sense of Section 2.3.2.1. Indicate with check-marks (✓) the diagrams that are intended to commute. If the 2-dimensionality of the page prevents a check-mark from being unambiguous, indicate the intended commutativity with an equation. ◊

Example 2.3.3.2 (Non-commuting diagram). In my conception of the world, the following diagram does not commute:

$$\begin{tikzcd}
\boxed{\text{a person}} \arrow[r, "\text{has as father}"] \arrow[dr, "\text{lives in}"'] & \boxed{\text{a man}} \arrow[d, "\text{lives in}"] \\
 & \boxed{\text{a city}}
\end{tikzcd} \qquad (2.19)$$

The non-commutativity of Diagram (2.19) does not imply that, in my conception, no person lives in the same city as his or her father. Rather it implies that, in my conception, it is not the case that *every* person lives in the same city as his or her father.

Exercise 2.3.3.3. Create an olog about a scientific subject, preferably one you think about often. The olog should have at least five boxes, five arrows, and one commutative diagram. ◊

2.3.3.4 A formula for writing facts as English

Every fact consists of two paths, say P and Q, that are to be declared equivalent. The paths P and Q will necessarily have the same source, say s, and target, say t, but their

[8] We defined function composition on page 2.1.2, but here we're using a different notation. There we would have said $g \circ f = i \circ h$, which is in the backwards-seeming *classical order*. Category theorists and others often prefer the *diagrammatic order* for writing compositions, which is $f; g = h; i$. For ologs, we follow the latter because it makes for better English sentences, and for the same reason we add the source object to the equation, writing $Afg \simeq Ahi$.

lengths may be different, say m and n respectively. [9] We draw these paths as

$$\begin{tikzcd}
P: & \overset{a_0=s}{\bullet} \arrow[r, "f_1"] & \overset{a_1}{\bullet} \arrow[r, "f_2"] & \overset{a_2}{\bullet} \arrow[r, "f_3"] & \cdots \arrow[r, "f_{m-1}"] & \overset{a_{m-1}}{\bullet} \arrow[r, "f_m"] & \overset{a_m=t}{\bullet} \\
Q: & \overset{b_0=s}{\bullet} \arrow[r, "g_1"] & \overset{b_1}{\bullet} \arrow[r, "g_2"] & \overset{b_2}{\bullet} \arrow[r, "g_3"] & \cdots \arrow[r, "g_{n-1}"] & \overset{b_{n-1}}{\bullet} \arrow[r, "g_n"] & \overset{b_n=t}{\bullet}
\end{tikzcd} \tag{2.20}$$

Every part ℓ of an olog (i.e. every box and every arrow) has an associated English phrase, which we write as "ℓ". Using a dummy variable x we can convert a fact into English too. The following general formula is a bit difficult to understand, see Example 2.3.3.5, but here goes. The fact $P \simeq Q$ from (2.20) can be Englishified as follows:

Given x, "s", consider the following. We know that x is "s", (2.21)
which "f_1" "a_1", which "f_2" "a_2", which ... "f_{m-1}" "a_{m-1}", which "f_m" "t"
that we'll call $P(x)$.
We also know that x is "s",
which "g_1" "b_1", which "g_2" "b_2", which ... "g_{n-1}" "b_{n-1}", which "g_n" "t"
that we'll call $Q(x)$.
Fact: whenever x is "s", we will have $P(x) = Q(x)$.

Example 2.3.3.5. Consider the olog

$$\begin{tikzcd}
\overset{A}{\boxed{\text{a person}}} \arrow[r, "\text{has}"] \arrow[dr, "\checkmark", "\text{lives in}"'] & \overset{B}{\boxed{\text{an address}}} \arrow[d, "\text{is in}"] \\
& \overset{C}{\boxed{\text{a city}}}
\end{tikzcd} \tag{2.22}$$

To put the fact that Diagram 2.22 commutes into English, we first Englishify the two paths: F="a person has an address which is in a city" and G="a person lives in a city". The source of both is s="a person" and the target of both is t="a city". write:

Given x, a person, consider the following. We know that x is a person,
which has an address, which is in a city
that we'll call $P(x)$.
We also know that x is a person,
which lives in a city
that we'll call $Q(x)$.
Fact: whenever x is a person, we will have $P(x) = Q(x)$.

[9] If the source equals the target, $s = t$, then it is possible to have $m = 0$ or $n = 0$, and the ideas below still make sense.

Exercise 2.3.3.6. This olog was taken from [Sp1].

$$\begin{tikzcd}
 & \overset{N}{\boxed{\text{a phone number}}} \arrow[r, "\text{has}"] & \overset{C}{\boxed{\text{an area code}}} \arrow[d, "\text{corresponds to}"] \\
\overset{OLP}{\boxed{\text{an operational land-line phone}}} \arrow[ur, "\text{is assigned}"] \arrow[r, "\text{is}"'] & \overset{P}{\boxed{\text{a physical phone}}} \arrow[r, "\text{is currently located in}"'] \arrow[ur, phantom, "\checkmark"] & \overset{R}{\boxed{\text{a region}}}
\end{tikzcd} \tag{2.23}$$

It says that a landline phone is physically located in the region that its phone number is assigned. Translate this fact into English using the formula from 2.21. ◊

Exercise 2.3.3.7. In the above olog (2.23), suppose that the box ⌜an operational landline phone⌝ is replaced with the box ⌜an operational mobile phone⌝. Would the diagram still commute? ◊

2.3.3.8 Images

In this section we discuss a specific kind of fact, generated by any aspect. Recall that every function has an image, meaning the subset of elements in the codomain that are "hit" by the function. For example the function $f(x) = 2 * x \colon \mathbb{Z} \to \mathbb{Z}$ has as image the set of all even numbers.

Similarly the set of mothers arises as is the image of the "has as mother" function, as shown below

$$\begin{tikzcd}
\overset{P}{\boxed{\text{a person}}} \arrow[rr, "\overset{f\colon P\to P}{\text{has as mother}}"] \arrow[dr, "\text{has}"'] & \checkmark & \overset{P}{\boxed{\text{a person}}} \\
 & \overset{M=\mathrm{im}(f)}{\boxed{\text{a mother}}} \arrow[ur, "\text{is}"'] &
\end{tikzcd}$$

Exercise 2.3.3.9. For each of the following types, write down a function for which it is the image, or say "not clearly an image type"

a.) ⌜a book⌝

b.) ⌜a material that has been fabricated by a process of type T⌝

c.) ⌜a bicycle owner⌝

d.) ⌜a child⌝

e.) ⌜a used book⌝

f.) ⌜an inhabited residence⌝

◊

2.4 Products and coproducts

In this section we introduce two concepts that are likely to be familiar, although perhaps not by their category-theoretic names, product and coproduct. Each is an example of a large class of ideas that exist far beyond the realm of sets.

2.4.1 Products

Definition 2.4.1.1. Let X and Y be sets. The *product of X and Y*, denoted $X \times Y$, is defined as the set of ordered pairs (x, y) where $x \in X$ and $y \in Y$. Symbolically,

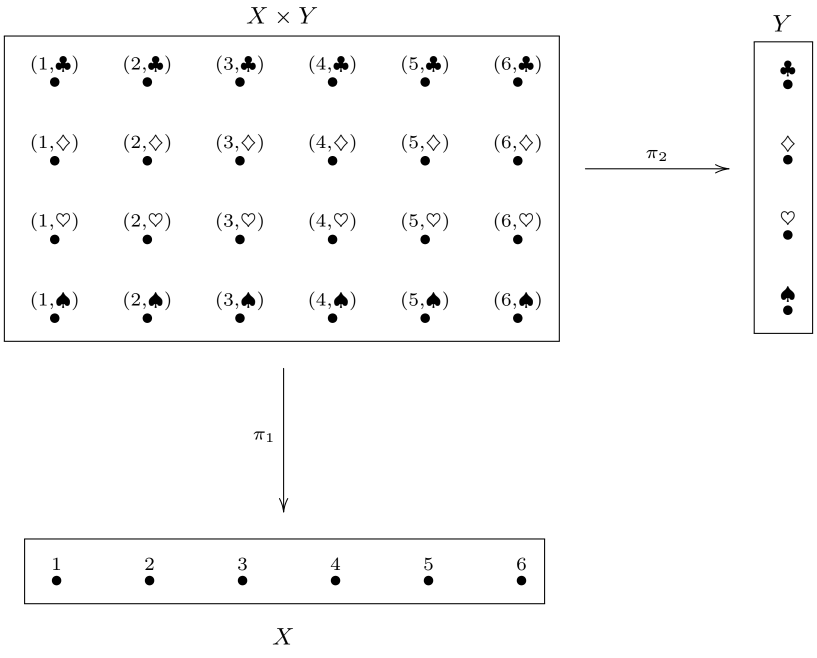
$$X \times Y = \{(x, y) \mid x \in X, \quad y \in Y\}.$$

There are two natural *projection functions* $\pi_1 \colon X \times Y \to X$ and $\pi_2 \colon X \times Y \to Y$.

$$\begin{tikzcd} & X \times Y \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & \\ X & & Y \end{tikzcd}$$

Example 2.4.1.2. [Grid of dots]

Let $X = \{1, 2, 3, 4, 5, 6\}$ and $Y = \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$. Then we can draw $X \times Y$ as a 6-by-4 grid of dots, and the projections as projections



(2.24)

Application 2.4.1.3. A traditional (Mendelian) way to predict the genotype of offspring based on the genotype of its parents is by the use of Punnett squares. If F is the set of possible genotypes for the female parent and M is the set of possible genotypes of the male parent, then $F \times M$ is drawn as a square, called a Punnett square, in which every combination is drawn. ◊◊

Exercise 2.4.1.4. How many elements does the set $\{a, b, c, d\} \times \{1, 2, 3\}$ have? ◊

Application 2.4.1.5. Suppose we are conducting experiments about the mechanical properties of materials, as in Application 2.1.2.1. For each material sample we will produce multiple data points in the set ⌜extension⌝ × ⌜force⌝ $\cong \mathbb{R} \times \mathbb{R}$.

◊◊

Remark 2.4.1.6. It is possible to take the product of more than two sets as well. For example, if A, B, and C are sets then $A \times B \times C$ is the set of triples,

$$A \times B \times C := \{(a, b, c) \mid a \in A, b \in B, c \in C\}.$$

This kind of generality is useful in understanding multiple dimensions, e.g. what physicists mean by 10-dimensional space. It comes under the heading of *limits*, which we will see in Section 4.5.3.

Example 2.4.1.7. Let $\mathbb{R}$ be the set of real numbers. By $\mathbb{R}^2$ we mean $\mathbb{R} \times \mathbb{R}$ (though see Exercise 2.7.2.6). Similarly, for any $n \in \mathbb{N}$, we define $\mathbb{R}^n$ to be the product of n copies of $\mathbb{R}$.

According to [Pen], Aristotle seems to have conceived of space as something like $S := \mathbb{R}^3$ and of time as something like $T := \mathbb{R}$. Spacetime, had he conceived of it, would probably have been $S \times T \cong \mathbb{R}^4$. He of course did not have access to this kind of abstraction, which was probably due to Descartes.

Exercise 2.4.1.8. Let $\mathbb{Z}$ denote the set of integers, and let $+\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ denote the addition function and $\cdot\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ denote the multiplication function. Which of the following diagrams commute?

a.)

$$\begin{tikzcd}
\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \arrow[r, "{(a,b,c) \mapsto (a \cdot b, a \cdot c)}"] \arrow[d, "{(a,b,c) \mapsto (a+b,c)}"'] & \mathbb{Z} \times \mathbb{Z} \arrow[d, "{(x,y) \mapsto x+y}"] \\
\mathbb{Z} \times \mathbb{Z} \arrow[r, "{(x,y) \mapsto xy}"'] & \mathbb{Z}
\end{tikzcd}$$

b.)

$$\begin{tikzcd}
\mathbb{Z} \arrow[r, "{x \mapsto (x,0)}"] \arrow[dr, "\mathrm{id}_{\mathbb{Z}}"'] & \mathbb{Z} \times \mathbb{Z} \arrow[d, "{(a,b) \mapsto a \cdot b}"] \\
 & \mathbb{Z}
\end{tikzcd}$$

c.)

$$\begin{tikzcd}
\mathbb{Z} \arrow[r, "{x \mapsto (x,1)}"] \arrow[dr, "\mathrm{id}_{\mathbb{Z}}"'] & \mathbb{Z} \times \mathbb{Z} \arrow[d, "{(a,b) \mapsto a \cdot b}"] \\
 & \mathbb{Z}
\end{tikzcd}$$

◊

2.4.1.9 Universal property for products

Lemma 2.4.1.10 (Universal property for product)**.** *Let X and Y be sets. For any set A and functions $f\colon A \to X$ and $g\colon A \to Y$, there exists a unique function $A \to X \times Y$*

such that the following diagram commutes [10]

$$\begin{tikzcd}
 & & X \times Y \arrow[dll, "\pi_1"'] \arrow[drr, "\pi_2"] & & \\
X & \checkmark & & \checkmark & Y \\
 & & A \arrow[ull, "\forall f"] \arrow[urr, "\forall g"'] \arrow[uu, dashed, "\exists!"] & &
\end{tikzcd} \tag{2.25}$$

We might write the unique function as

$$\langle f, g \rangle \colon A \to X \times Y.$$

Proof. Suppose given f, g as above. To provide a function $\ell \colon A \to X \times Y$ is equivalent to providing an element $\ell(a) \in X \times Y$ for each $a \in A$. We need such a function for which $\pi_1 \circ \ell = f$ and $\pi_2 \circ \ell = g$. An element of $X \times Y$ is an ordered pair (x, y), and we can use $\ell(a) = (x, y)$ if and only if $x = \pi_1(x, y) = f(a)$ and $y = \pi_2(x, y) = g(a)$. So it is necessary and sufficient to define

$$\langle f, g \rangle(a) := (f(a), g(a))$$

for all $a \in A$.

□

Example 2.4.1.11 (Grid of dots, continued). We need to see the universal property of products as completely intuitive. Recall that if X and Y are sets, say of cardinalities $|X| = m$ and $|Y| = n$ respectively, then $X \times Y$ is an $m \times n$ grid of dots, and it comes with two canonical projections $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$. These allow us to extract from every grid element $z \in X \times Y$ its column $\pi_1(z) \in X$ and its row $\pi_2(z) \in Y$.

Suppose that each person in a classroom picks an element of X and an element of Y. Thus we have functions $f \colon C \to X$ and $g \colon C \to Y$. But isn't picking a column and a row the same thing as picking an element in the grid? The two functions f and g induce a unique function $C \to X \times Y$. And how does this function $C \to X \times Y$ compare with the original functions f and g? The commutative diagram (2.25) sums up the obvious connection.

Example 2.4.1.12. Let $\mathbb{R}$ be the set of real numbers. The origin in $\mathbb{R}$ is an element of $\mathbb{R}$. As you showed in Exercise 2.1.2.14, we can view this (or any) element of $\mathbb{R}$ as a function $z \colon \{☺\} \to \mathbb{R}$, where $\{☺\}$ is any set with one element. Our function z "picks out the origin". Thus we can draw functions

$$\begin{tikzcd}
 & \{☺\} \arrow[dl, "z"'] \arrow[dr, "z"] & \\
\mathbb{R} & & \mathbb{R}
\end{tikzcd}$$

[10] The symbol $\forall$ is read "for all"; the symbol $\exists$ is read "there exists", and the symbol $\exists!$ is read "there exists a unique". So this diagram is intended to express the idea that for any functions $f \colon A \to X$ and $g \colon A \to Y$, there exists a unique function $A \to X \times Y$ for which the two triangles commute.

The universal property for products guarantees a function $\{\text{☺}\} \to \mathbb{R} \times \mathbb{R}$, which will be the origin in $\mathbb{R}^2$.

Remark 2.4.1.13. Given sets X, Y, and A, and functions $f\colon A \to X$ and $g\colon A \to Y$, there is a unique function $A \to X \times Y$ that commutes with f and g. We call it *the induced function* $A \to X \times Y$, meaning the one that arises in light of f and g.

Exercise 2.4.1.14. For every set A there is some nice relationship between the following three sets:

$$\mathrm{Hom}_{\mathbf{Set}}(A, X), \qquad \mathrm{Hom}_{\mathbf{Set}}(A, Y), \qquad \text{and} \qquad \mathrm{Hom}_{\mathbf{Set}}(A, X \times Y).$$

What is it?

Hint: Do not be alarmed: this problem is a bit "recursive" in that you'll use products in your formula. ◊

Exercise 2.4.1.15.

a.) Let X and Y be sets. Construct the "swap map" $s\colon X \times Y \to Y \times X$ using only the universal property for products. If $\pi_1\colon X \times Y \to X$ and $\pi_2\colon X \times Y \to Y$ are the projection functions, write s in terms of the symbols "π_1", "π_2", "$(\, , \,)$", and "$\circ$".

b.) Can you prove that s is a isomorphism using only the universal property for product?

◊

Example 2.4.1.16. Suppose given sets X, X', Y, Y' and functions $m\colon X \to X'$ and $n\colon Y \to Y'$. We can use the universal property of products to construct a function $s\colon X \times Y \to X' \times Y'$. Here's how.

The universal property (Lemma 2.4.1.10) says that to get a function from any set A to $X' \times Y'$, we need two functions, namely some $f\colon A \to X'$ and some $g\colon A \to Y'$. Here $A = X \times Y$.

What we have readily available are the two projections $\pi_1\colon X \times Y \to X$ and $\pi_2\colon X \times Y \to Y$. But we also have $m\colon X \to X'$ and $n\colon Y \to Y'$. Composing, we set $f := m \circ \pi_1$ and $g := n \circ \pi_2$.

$$\begin{tikzcd}
 & X' \times Y' \arrow[dl, "\pi'_1"'] \arrow[dr, "\pi'_2"] & \\
X' & & Y' \\
X \arrow[u, "m"] & & Y \arrow[u, "n"'] \\
 & X \times Y \arrow[ul, "\pi_1"] \arrow[ur, "\pi_2"'] \arrow[uuu, dashed] &
\end{tikzcd}$$

The dotted arrow is often called the *product* of $m\colon X \to X'$ and $n\colon Y \to Y'$ and is denoted simply by

$$m \times n\colon X \times Y \to X' \times Y'.$$

2.4.1.17 Ologging products

Given two objects c, d in an olog, there is a canonical label "$c \times d$" for their product $c \times d$, written in terms of the labels "c" and "d". Namely,

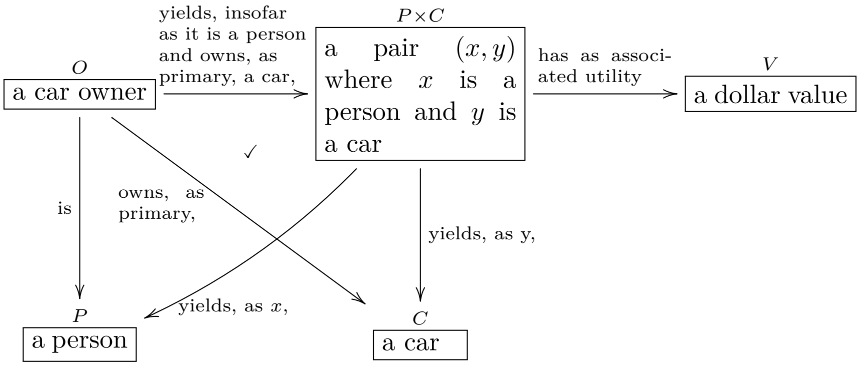
$$\text{“}c \times d\text{”} := \text{a pair } (x, y) \text{ where } x \text{ is “}c\text{” and } y \text{ is “}d\text{”.}$$

The projections $c \leftarrow c \times d \rightarrow d$ can be labeled "yields, as x," and "yields, as y," respectively.

Suppose that e is another object and $p\colon e \rightarrow c$ and $q\colon e \rightarrow d$ are two arrows. By the universal property of products (Lemma 2.4.1.10), p and q induce a unique arrow $e \rightarrow c \times d$ making the evident diagrams commute. This arrow can be labeled

yields, insofar as it "p" "c" and "q" "d",

Example 2.4.1.18. Every car owner owns at least one car, but there is no obvious function ⌜a car owner⌝ → ⌜a car⌝ because he or she may own more than one. One good choice would be the car that the person drives most often, which we'll call his or her primary car. Also, given a person and a car, an economist could ask how much utility the person would get out of the car. From all this we can put together the following olog involving products:


yields, insofar as it is a person and owns, as primary, a car,
$P\times C$
a pair (x,y) where x is a person and y is a car
has as associated utility
V
a dollar value
O
a car owner
✓
is
owns, as primary,
yields, as y,
yields, as x,
P
a person
C
a car

2.4.2 Coproducts

Definition 2.4.2.1. Let X and Y be sets. The *coproduct of* X *and* Y, denoted $X \sqcup Y$, is defined as the "disjoint union" of X and Y, i.e. the set for which an element is either an element of X or an element of Y. If something is an element of both X and Y then we include both copies, and distinguish between them, in $X \sqcup Y$. See Example 2.4.2.2

There are two natural inclusion functions $i_1\colon X \rightarrow X \sqcup Y$ and $i_2\colon Y \rightarrow X \sqcup Y$.

$$\begin{tikzcd}
X \arrow[dr, "i_1"'] & & Y \arrow[dl, "i_2"] \\
& X \sqcup Y &
\end{tikzcd}$$

Example 2.4.2.2. The coproduct of $X := \{a, b, c, d\}$ and $Y := \{1, 2, 3\}$ is

$$X \sqcup Y \cong \{a, b, c, d, 1, 2, 3\}.$$

The coproduct of X and itself is

$$X \sqcup X \cong \{i_1a, i_1b, i_1c, i_1d, i_2a, i_2b, i_2c, i_2d\}$$

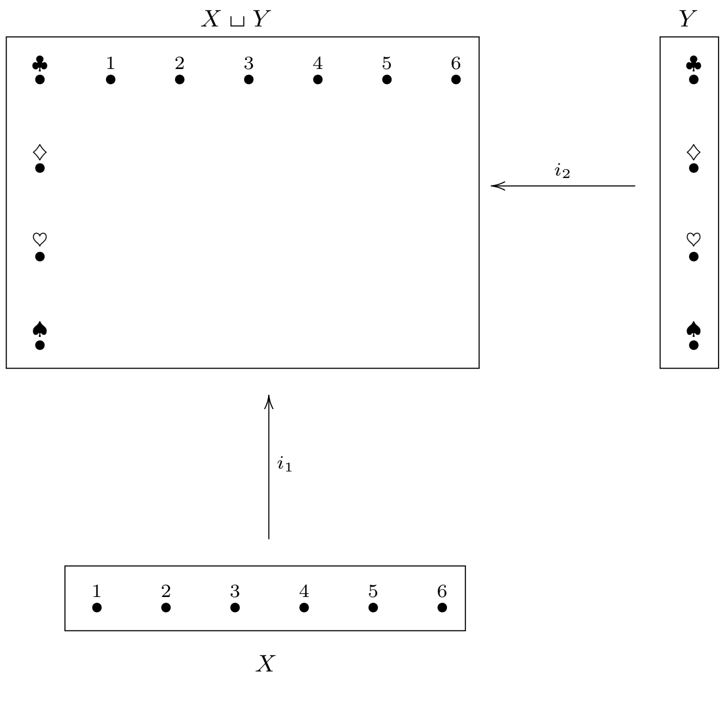
The names of the elements in $X \sqcup Y$ are not so important. What's important are the inclusion maps i_1, i_2, which ensure that we know where each element of $X \sqcup Y$ came from.

Example 2.4.2.3 (Airplane seats).

$$\begin{tikzcd}
\overset{X}{\boxed{\text{an economy-class seat in an airplane}}} \arrow[dr, "\text{is}"'] & & \overset{Y}{\boxed{\text{a first-class seat in an airplane}}} \arrow[dl, "\text{is}"] \\
& \overset{X \sqcup Y}{\boxed{\text{a seat in an airplane}}} &
\end{tikzcd} \tag{2.26}$$

Exercise 2.4.2.4. Would you say that ⌜a phone⌝ is the coproduct of ⌜a cellphone⌝ and ⌜a landline phone⌝? ◊

Example 2.4.2.5 (Disjoint union of dots).



(2.27)

2.4.2.6 Universal property for coproducts

Lemma 2.4.2.7 (Universal property for coproduct)**.** *Let X and Y be sets. For any set A and functions $f\colon X \to A$ and $g\colon Y \to A$, there exists a unique function $X \sqcup Y \to A$*

such that the following diagram commutes

$$\begin{tikzcd}
& A & \\
X \arrow[ur, "\forall f"] \arrow[dr, "i_1"'] & & Y \arrow[ul, "\forall g"'] \arrow[dl, "i_2"] \\
& X \sqcup Y \arrow[uu, dashed, "\exists!"] &
\end{tikzcd}$$

We might write the unique function as [11]

$$\begin{cases} f \\ g \end{cases} : X \sqcup Y \to A.$$

Proof. Suppose given f, g as above. To provide a function $\ell\colon X \sqcup Y \to A$ is equivalent to providing an element $f(m) \in A$ is for each $m \in X \sqcup Y$. We need such a function such that $\ell \circ i_1 = f$ and $\ell \circ i_2 = g$. But each element $m \in X \sqcup Y$ is either of the form $i_1 x$ or $i_2 y$, and cannot be of both forms. So we assign

$$\begin{cases} \mathrm{f} \\ \mathrm{g} \end{cases} (m) = \begin{cases} f(x) & \text{if } m = i_1 x, \\ g(y) & \text{if } m = i_2 y. \end{cases}$$

This assignment is necessary and sufficient to make all relevant diagrams commute.

□

Example 2.4.2.8 (Airplane seats, continued). The universal property of coproducts says the following. Any time we have a function $X \to A$ and a function $Y \to A$, we get a unique function $X \sqcup Y \to A$. For example, every economy class seat in an airplane and every first class seat in an airplane is actually *in a particular airplane.* Every economy class seat has a price, as does every first class seat.

$$\begin{tikzcd}
& & \overset{A}{\boxed{\text{a dollar figure}}} & & \\
& \checkmark & & \checkmark & \\
\overset{X}{\boxed{\text{an economy-class seat in an airplane}}} \arrow[uurr, "\text{has as price}"] \arrow[rr, "\text{is}"] \arrow[ddrr, "\text{is in}"'] & & \overset{X \sqcup Y}{\boxed{\text{a seat in an airplane}}} \arrow[uu, dashed, "\exists!"] \arrow[dd, dashed, "\exists!"'] & & \overset{Y}{\boxed{\text{a first-class seat in an airplane}}} \arrow[uull, "\text{has as price}"'] \arrow[ll, "\text{is}"'] \arrow[ddll, "\text{is in}"] \\
& \checkmark & & \checkmark & \\
& & \overset{B}{\boxed{\text{an airplane}}} & &
\end{tikzcd} \tag{2.28}$$

The universal property of coproducts formalizes the following intuitively obvious fact:

[11] We are about to use a two-line symbol, which is a bit unusual. In what follows a certain function $X \sqcup Y \to A$ is being denoted by the symbol $\begin{cases} \mathrm{f} \\ \mathrm{g} \end{cases}$.

> If we know how economy class seats are priced and we know how first class seats are priced, and if we know that every seat is either economy class or first class, then we automatically know how all seats are priced.

To say it another way (and using the other induced map):

> If we keep track of which airplane every economy class seat is in and we keep track of which airplane every first class seat is in, and if we know that every seat is either economy class or first class, then we require no additional tracking for any airplane seat whatsoever.

Application 2.4.2.9 (Piecewise defined curves). In science, curves are often defined or considered piecewise. For example in testing the mechanical properties of a material, we might be interested in various regions of deformation, such as elastic, plastic, or post-fracture. These are three intervals on which the material displays different kinds of properties.

For real numbers $a < b \in \mathbb{R}$, let $[a,b] := \{x \in \mathbb{R} \mid a \leqslant x \leqslant b\}$ denote the closed interval. Given a function $[a,b] \to \mathbb{R}$ and a function $[c,d] \to \mathbb{R}$, the universal property of coproducts implies that they extend uniquely to a function $[a,b] \sqcup [c,d] \to \mathbb{R}$, which will appear as a piecewise defined curve.

Often we are given a curve on $[a,b]$ and another on $[b,c]$, where the two curves agree at the point b. This situation is described by pushouts, which are mild generalizations of coproducts; see Section 2.6.2.

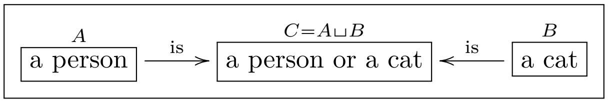
◊◊

Exercise 2.4.2.10. Write the universal property for coproduct in terms of a relationship between the following three sets:

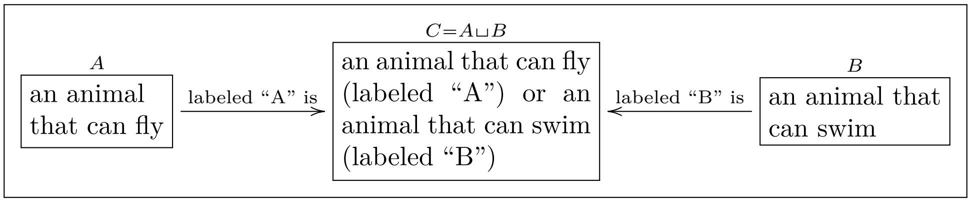
$$\mathrm{Hom}_{\mathbf{Set}}(X,A), \qquad \mathrm{Hom}_{\mathbf{Set}}(Y,A), \qquad \text{and} \qquad \mathrm{Hom}_{\mathbf{Set}}(X \sqcup Y, A).$$

◊

Example 2.4.2.11. In the following olog the types A and B are disjoint, so the coproduct $C = A \sqcup B$ is just the union.



Example 2.4.2.12. In the following olog, A and B are not disjoint, so care must be taken to differentiate common elements.



Since ducks can both swim and fly, each duck is found twice in C, once labeled as a flyer and once labeled as a swimmer. The types A and B are kept disjoint in C, which justifies the name "disjoint union."

Exercise 2.4.2.13. Understand Example 2.4.2.12 and see if a similar idea would make sense for particles and waves. Make an olog, and choose your wording in accordance with Rules 2.3.1.2. How do photons, which exhibit properties of both waves and particles, fit into the coproduct in your olog?

◊

Exercise 2.4.2.14. Following the section above, "Ologging products" page 36, come up with a naming system for coproducts, the inclusions, and the universal maps. Try it out by making an olog (involving coproducts) discussing the idea that both a .wav file and a .mp3 file can be played on a modern computer. Be careful that your arrows are valid in the sense of Section 2.3.2.1. ◊

2.5 Finite limits in Set

In this section we discuss what are called *limits* of variously-shaped diagrams of sets. We will make all this much more precise when we discuss limits in arbitrary categories in Section 4.5.3.

2.5.1 Pullbacks

Definition 2.5.1.1 (Pullback)**.** Suppose given the diagram of sets and functions below.

$$\begin{tikzcd} & Y \arrow[d, "g"] \\ X \arrow[r, "f"'] & Z \end{tikzcd} \tag{2.29}$$

Its *fiber product* is the set

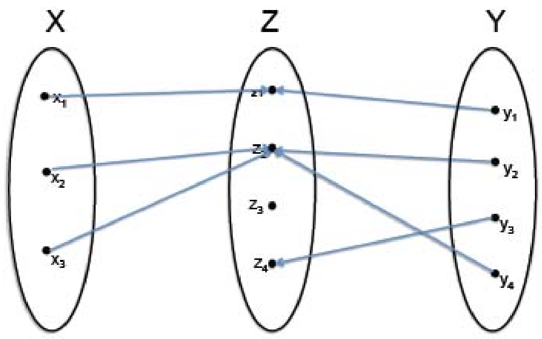
$$X \times_Z Y := \{(x, w, y) \mid f(x) = w = g(y)\}.$$

There are obvious projections $\pi_1 \colon X \times_Z Y \to X$ and $\pi_2 \colon X \times_Z Y \to Y$ (e.g. $\pi_2(x, w, y) = y$). Note that if $W = X \times_Z Y$ then the diagram

$$\begin{tikzcd} W \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] \arrow[dr, phantom, very near start, "\lrcorner"] & Y \arrow[d, "g"] \\ X \arrow[r, "f"'] & Z \end{tikzcd} \tag{2.30}$$

commutes. Given the setup of Diagram 2.29 we define the *pullback of X and Y over Z* to be any set W for which we have an isomorphism $W \xrightarrow{\cong} X \times_Z Y$. The corner symbol $\lrcorner$ in Diagram 2.30 indicates that W is the pullback.

Exercise 2.5.1.2. Let X, Y, Z be as drawn and $f \colon X \to Z$ and $g \colon Y \to Z$ the indicated functions.



What is the pullback of the diagram $X \xrightarrow{f} Z \xleftarrow{g} Y$? ◊

Exercise 2.5.1.3.

a.) Draw a set X with five elements and a set Y with three elements. Color each element of X and each element of Y either red, blue, or yellow, [12] and do so in a "random-looking" way. Considering your coloring of X as a function $X \to C$, where $C = \{\text{red, blue, yellow}\}$, and similarly obtaining a function $Y \to C$, draw the fiber product $X \times_C Y$. Make sure it is colored appropriately.

b.) The universal property for products guarantees a function $X \times_C Y \to X \times Y$, which I can tell you will be an injection. This means that the drawing you made of the fiber product can be imbedded into the 5×3 grid; please draw the grid and indicate this subset.

◊

Remark 2.5.1.4. Some may prefer to denote this fiber product by $f \times_Z g$ rather than $X \times_Z Y$. The former is mathematically better notation, but human-readability is often enhanced by the latter, which is also more common in the literature. We use whichever is more convenient.

Exercise 2.5.1.5.

a.) Suppose that $Y = \varnothing$; what can you say about $X \times_Z Y$?

b.) Suppose now that Y is any set but that Z has exactly one element; what can you say about $X \times_Z Y$?

◊

Exercise 2.5.1.6. Let $S = \mathbb{R}^3, T = \mathbb{R}$, and think of them as (Aristotelian) space and time, with the origin in $S \times T$ given by the center of mass of MIT at the time of its founding. Let $Y = S \times T$ and let $g_1 \colon Y \to S$ be one projection and $g_2 \colon Y \to T$ the other projection. Let $X = \{☺\}$ be a set with one element and let $f_1 \colon X \to S$ and $f_2 \colon X \to T$ be given by the origin in both cases.

a.) What are the fiber products W_1 and W_2:

$$\begin{tikzcd}
W_1 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[d, "g_1"] \\
X \arrow[r, "f_1"'] & S
\end{tikzcd}
\qquad
\begin{tikzcd}
W_2 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[d, "g_2"] \\
X \arrow[r, "f_2"'] & T
\end{tikzcd}$$

[12] You can use shadings rather than coloring, if coloring would be annoying.

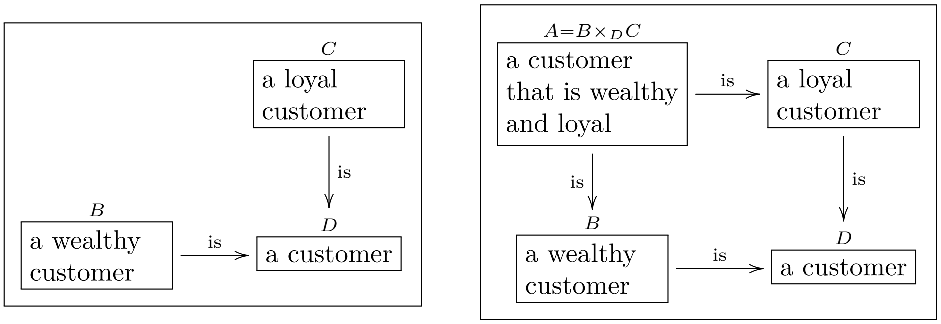
b.) Interpret these sets in terms of the center of mass of MIT at the time of its founding.

◊

2.5.1.7 Using pullbacks to define new ideas from old

In this section we will see that the fiber product of a diagram can serve to define a new concept. For example, in (2.33) we define what it means for a cellphone to have a bad battery, in terms of the length of time for which it remains charged. By being explicit, we reduce the chance of misunderstandings between different groups of people. This can be useful in situations like audits and those in which one is trying to reuse or understand data gathered by others.

Example 2.5.1.8. Consider the following two ologs. The one on the right is the pullback of the one on the left.



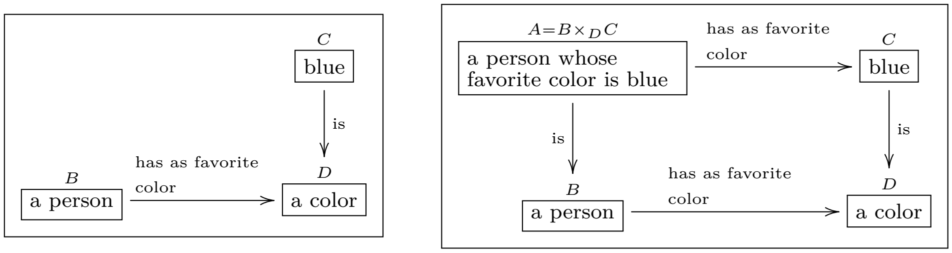
(2.31)

Check from Definition 2.5.1.1 that the label, "a customer that is wealthy and loyal", is fair and straightforward as a label for the fiber product $A = B \times_D C$, given the labels on B, C, and D.

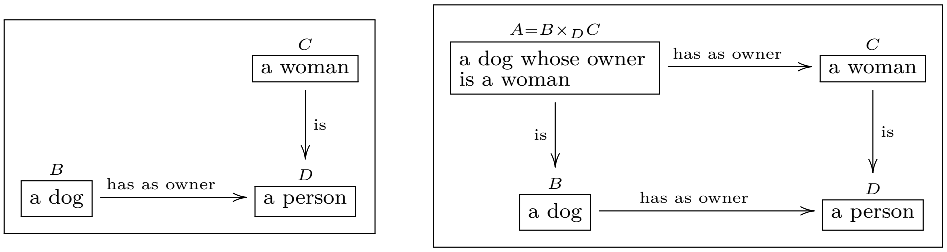
Remark 2.5.1.9. Note that in Diagram (2.31) the top-left box could have been (non-canonically named) ⌜a good customer⌝. If it was taken to be the fiber product, then the author would be effectively *defining* a good customer to be one that is wealthy and loyal.

Exercise 2.5.1.10. For each of the following, an author has proposed that the diagram on the right is a pullback. Do you think their labels are appropriate or misleading; that is, is the label on the upper-left box reasonable given the rest of the olog, or is it suspect in some way?

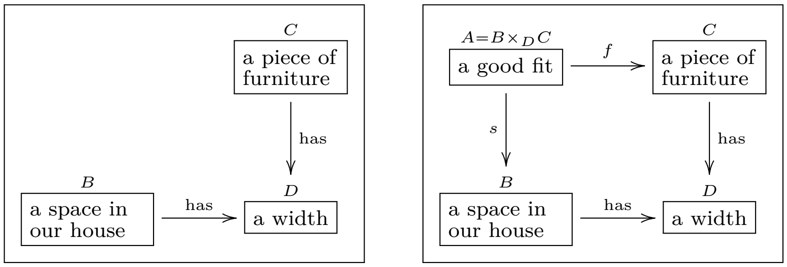
a.)



b.)



c.)



◊

Exercise 2.5.1.11.

a.) Consider your olog from Exercise 2.3.3.1. Are any of the commutative squares there actually pullback squares?

b.) Now use ologs with products and pullbacks to define what a brother is and what a sister is (again in a human biological nuclear family), in terms of types such as ⌜an offspring of mating pair (a, b)⌝, ⌜a person⌝, ⌜a male person⌝, ⌜a female person⌝, and so on.

◊

Definition 2.5.1.12 (Preimage)**.** Let $f\colon X \to Y$ be a function and $y \in Y$ an element. The *preimage of y under f*, denoted $f^{-1}(y)$, is the subset $f^{-1}(y) := \{x \in X \mid f(x) = y\}$. If $Y' \subseteq Y$ is any subset, the *preimage of Y' under f*, denoted $f^{-1}(Y')$, is the subset $f^{-1}(Y') = \{x \in X \mid f(x) \in Y'\}$.

Exercise 2.5.1.13. Let $f\colon X \to Y$ be a function and $y \in Y$ an element. Draw a pullback diagram in which the fiber product is isomorphic to the preimage $f^{-1}(y)$. ◊

Lemma 2.5.1.14 (Universal property for pullback)**.** *Suppose given the diagram of sets and functions as below.*

$$\begin{tikzcd}
 & Y \arrow[d, "u"] \\
X \arrow[r, "t"'] & Z
\end{tikzcd}$$

For any set A and commutative solid arrow diagram as below (i.e. functions $f\colon A \to X$ and $g\colon A \to Y$ such that $t \circ f = u \circ g$),

$$\begin{tikzcd}
& X \times_Z Y \arrow[ddl, bend right, "\pi_1"'] \arrow[ddr, bend left, "\pi_2"] & \\
& A \arrow[u, dashed, "\exists!"] \arrow[dl, "\forall f"'] \arrow[dr, "\forall g"] & \\
X \arrow[dr, "t"'] & & Y \arrow[dl, "u"] \\
& Z &
\end{tikzcd} \tag{2.32}$$

there exists a unique arrow $\langle f, f\rangle_Z\colon A \to X \times_Z Y$ making everything commute, i.e.

$$f = \pi_1 \circ \langle f, f\rangle_Z \qquad \textit{and} \qquad g = \pi_2 \circ \langle f, f\rangle_Z.$$

Exercise 2.5.1.15. Create an olog whose underlying shape is a commutative square. Now add the fiber product so that the shape is the same as that of Diagram (2.32). Assign English labels to the projections π_1, π_2 and to the dotted map $A \xrightarrow{\langle f,f\rangle_Z} X \times_Z Y$, such that these labels are as canonical as possible. ◊

2.5.1.16 Pasting diagrams for pullback

Consider the diagram drawn below, which includes a left-hand square, a right-hand square, and a big rectangle.

$$\begin{tikzcd}
A' \arrow[r, "f'"] \arrow[d, "i"'] \arrow[dr, phantom, "\lrcorner", very near start] & B' \arrow[r, "g'"] \arrow[d, "j"'] \arrow[dr, phantom, "\lrcorner", very near start] & C' \arrow[d, "k"] \\
A \arrow[r, "f"'] & B \arrow[r, "g"'] & C
\end{tikzcd}$$

The right-hand square has a corner symbol indicating that $B' \cong B \times_C C'$ is a pullback. But the corner symbol on the left is ambiguous; it might be indicating that the left-hand square is a pullback, or it might be indicating that the big rectangle is a pullback. It turns out that if $B' \cong B \times_C C'$ then it is not ambiguous because the left-hand square is a pullback if and only if the big rectangle is.

Proposition 2.5.1.17. *Consider the diagram drawn below*

$$\begin{tikzcd}
& B' \arrow[r, "g'"] \arrow[d, "j"'] \arrow[dr, phantom, "\lrcorner", very near start] & C' \arrow[d, "k"] \\
A \arrow[r, "f"'] & B \arrow[r, "g"'] & C
\end{tikzcd}$$

where $B' \cong B \times_C C'$ is a pullback. Then there is an isomorphism $A \times_B B' \cong A \times_C C'$. Said another way,

$$A \times_B (B \times_C C') \cong A \times_C C'.$$

Proof. We first provide a map $\phi\colon A\times_B(B\times_C C')\to A\times_C C'$. An element of $A\times_B(B\times_C C')$ is of the form $(a,b,(b,c,c'))$ such that $f(a)=b, g(b)=c$ and $k(c')=c$. But this implies that $g\circ f(a)=c=k(c')$ so we put $\phi(a,b,(b,c,c')):=(a,c,c')\in A\times_C C'$. Now we provide a proposed inverse, $\psi\colon A\times_C C'\to A\times_B(B\times_C C')$. Given (a,c,c') with $g\circ f(a)=c=k(c')$, let $b=f(a)$ and note that (b,c,c') is an element of $B\times_C C'$. So we can define $\psi(a,c,c')=(a,b,(b,c,c'))$. It is easy to see that ϕ and ψ are inverse. □

Proposition 2.5.1.17 can be useful in authoring ologs. For example, the type ⌜a cellphone that has a bad battery⌝ is vague, but we can lay out precisely what it means using pullbacks:

$$\begin{tikzcd}
\overset{A\cong B\times_D C}{\boxed{\text{a cellphone that has a bad battery}}} \arrow[r] \arrow[d] & \overset{C\cong D\times_F E}{\boxed{\text{a bad battery}}} \arrow[r] \arrow[d] & \overset{E\cong F\times_H G}{\boxed{\text{less than 1 hour}}} \arrow[r] \arrow[d] & \overset{G}{\boxed{\text{between 0 and 1}}} \arrow[d] \\
\overset{B}{\boxed{\text{a cellphone}}} \arrow[r, "\text{has}"] & \overset{D}{\boxed{\text{a battery}}} \arrow[r, "\text{remains charged for}"] & \overset{F}{\boxed{\text{a duration of time}}} \arrow[r, "\text{in hours yields}"] & \overset{H}{\boxed{\text{a range of numbers}}}
\end{tikzcd}$$

(2.33)

The category-theoretic fact described above says that since $A\cong B\times_D C$ and $C\cong D\times_F E$, it follows that $A\cong B\times_F E$. That is, we can deduce the definition "a cellphone that has a bad battery is defined as a cellphone that has a battery which remains charged for less than one hour."

Exercise 2.5.1.18.

a.) Create an olog that defines two people to be "of approximately the same height" if and only if their height difference is less than half an inch, using a pullback. Your olog can include the box ⌜a real number x such that $-.5<x<.5$⌝.

b.) In the same olog, make a box for those people whose height is approximately the same as a person named "The Virgin Mary". You may need to use images, as in Section 2.3.3.8.

◊

Exercise 2.5.1.19. Consider the diagram on the left below, where both squares commute.

$$\begin{tikzcd}
 & & & Y' \arrow[dd] \\
 & & Y \arrow[ur] \arrow[dd] & \\
 & X' \arrow[rr] & & Z' \\
X \arrow[ur] \arrow[rr] & & Z \arrow[ur] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & W' \arrow[rr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & Y' \arrow[dd] \\
W \arrow[rr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & Y \arrow[ur] \arrow[dd] & \\
 & X' \arrow[rr] & & Z' \\
X \arrow[ur] \arrow[rr] & & Z \arrow[ur] &
\end{tikzcd}$$

Let $W=X\times_Z Y$ and $W'=X'\times_{Z'}Y'$, and form the diagram to the right. Use the universal property of fiber products to construct a map $W\to W'$ such that all squares commute. ◊

2.5.2 Spans, experiments, and matrices

Definition 2.5.2.1. Given sets A and B, a *span on A and B* is a set R together with functions $f\colon R \to A$ and $g\colon R \to B$.

$$\begin{tikzcd} & R \arrow[dl, "f"'] \arrow[dr, "g"] & \\ A & & B \end{tikzcd}$$

Application 2.5.2.2. Think of A and B as observables and R as a set of experiments performed on these two variables. For example, let's say T is the set of possible temperatures of a gas in a fixed container and let's say P is the set of possible pressures of the gas. We perform 1000 experiments in which we change and record the temperature and we simultaneously also record the pressure; this is a span $T \xleftarrow{f} E \xrightarrow{g} P$. The results might look like this:

Experiment		
ID	**Temperature**	**Pressure**
1	100	72
2	100	73
3	100	72
4	200	140
5	200	138
6	200	141
⋮	⋮	⋮

◊◊

Definition 2.5.2.3. Let A, B, and C be sets, and let $A \xleftarrow{f} R \xrightarrow{g} B$ and $B \xleftarrow{f'} R' \xrightarrow{g'} C$ be spans. Their *composite span* is given by the fiber product $R \times_B R'$ as in the diagram below:

$$\begin{tikzcd} & & R \times_B R' \arrow[dl] \arrow[dr] & & \\ & R \arrow[dl, "f"'] \arrow[dr, "g"] & & R' \arrow[dl, "f'"'] \arrow[dr, "g'"] & \\ A & & B & & C \end{tikzcd}$$

Application 2.5.2.4. Let's look back at our lab's experiment from Application 2.5.2.2, which resulted in a span $T \xleftarrow{f} E \xrightarrow{g} P$. Suppose we notice that something looks a little wrong. The pressure should be linear in the temperature but it doesn't appear to be. We hypothesize that the volume of the container is increasing under pressure. We look up this container online and see that experiments have been done to measure the volume as the interior pressure changes. The data has generously been made available online, which gives us a span $P \xleftarrow{f'} E' \xrightarrow{g'} V$.

The composite of our lab's span with the online data span yields a span $T \leftarrow E'' \to V$, where $E'' := E \times_P E'$. What information does this span give us? In explaining it, one might say “whenever an experiment in our lab yielded the same pressure as one they recorded, let’s call that a data point. Every data point has an associated temperature (from our lab) and an associated volume (from their experiment). This is the best we can do.”

The information we get this way might be seen by some as unscientific, but it certainly is the kind of information people use in business and in every day life calculation—we get our data from multiple sources and put it together. Moreover, it is scientific in the sense that it is reproducible. The way we obtained our T-V data is completely transparent.

◊◊

We can relate spans to matrices of natural numbers, and see a natural “categorification” of matrix addition and matrix multiplication. If our spans come from experiments as in Applications 2.5.2.2 and 2.5.2.4 the matrices involved will look like huge but sparse matrices. Let’s go through that.

Let A and B be sets and let $A \leftarrow R \rightarrow B$ be a span. By the universal property of products, we have a unique map $R \xrightarrow{p} A \times B$.

We make a matrix of natural numbers out of this data as follows. The set of rows is A, the set of columns is B. For elements $a \in A$ and $b \in B$, the (a, b)-entry is the cardinality of its preimage, $|p^{-1}(a, b)|$, i.e. the number of elements in R that are sent by p to (a, b).

Suppose we are given two (A, B)-spans, i.e. $A \leftarrow R \rightarrow B$ and $A \leftarrow R' \rightarrow B$; we might think of these has having the same *dimensions*, i.e. they are both $|A| \times |B|$-matrices. We can take the disjoint union $R \sqcup R'$ and by the universal property of coproducts we have a unique span $A \leftarrow R \sqcup R' \rightarrow B$ making the requisite diagram commute. [13] The matrix corresponding to this new span will be the sum of the matrices corresponding to the two previous spans out of which it was made.

Given a span $A \leftarrow R \rightarrow B$ and a span $B \leftarrow S \rightarrow C$, the composite span can be formed as in Definition 2.5.2.3. It will correspond to the usual multiplication of matrices.

Construction 2.5.2.5. Given a span $A \xleftarrow{f} R \xrightarrow{g} B$, one can draw a *bipartite graph* with each element of A drawn as a dot on the left, each element of B drawn as a dot on the right, and each element $r \in R$ drawn as an arrow connecting vertex $f(r)$ on the left to vertex $g(r)$ on the right.

Exercise 2.5.2.6.

a.) Draw the bipartite graph (as in Construction 2.5.2.5) corresponding to the span $T \xleftarrow{f} E \xrightarrow{g} P$ in Application 2.5.2.2.

b.) Now make up your own span $P \xleftarrow{f'} E' \xrightarrow{g'} V$ and draw it. Finally, draw the composite span below.

c.) Can you say how the composite span graph relates to the graphs of its factors?

[13]
$$\begin{tikzcd}
 & R \arrow[dl] \arrow[d] \arrow[dr] & \\
A & R \sqcup R' \arrow[l] \arrow[r] & B \\
 & R' \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

◊

2.5.3 Equalizers and terminal objects

Definition 2.5.3.1. Suppose given two parallel arrows

$$\begin{tikzcd} X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y. \end{tikzcd} \qquad \begin{tikzcd} Eq(f,g) \arrow[r, "p"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \end{tikzcd} \tag{2.34}$$

The *equalizer of f and g* is the commutative diagram as to the right in (2.34), where we define

$$Eq(f,g) := \{x \in X \mid f(x) = g(x)\}$$

and where p is the canonical inclusion.

Example 2.5.3.2. Suppose one has designed an experiment to test a theoretical prediction. The question becomes, "when does the theory match the experiment?" The answer is given by the equalizer of the following diagram:

$$\begin{tikzcd} \boxed{\text{an input}} \arrow[r, shift left, "\text{should, according to theory, yield}"] \arrow[r, shift right, "\text{according to experiment yields}"'] & \boxed{\text{an output}} \end{tikzcd}$$

The equalizer is the set of all inputs for which the theory and the experiment yield the same output.

Exercise 2.5.3.3. Come up with an olog that uses equalizers in a reasonably interesting way. Alternatively, use an equalizer to specify those published authors who have published exactly one paper. Hint: find a function from authors to papers; then find another. ◊

Exercise 2.5.3.4. Find a universal property enjoyed by the equalizer of two arrows, and present it in the style of Lemmas 2.4.1.10, 2.4.2.7, and 2.5.1.14. ◊

Exercise 2.5.3.5.

a.) A terminal set is a set S such that for every set X, there exists a unique function $X \to S$. Find a terminal set.

b.) Do you think that the notion *terminal set* belongs in this section (Section 2.5)? How so? If products, pullbacks, and equalizers are all limits, what do limits have in common?

◊

2.6 Finite colimits in Set

This section will parallel Section 2.5—I will introduce several types of finite colimits and hope that this gives the reader some intuition about them, without formally defining them yet. Before doing so, I must define equivalence relations and quotients.

2.6.1 Background: equivalence relations

Definition 2.6.1.1 (Equivalence relations and equivalence classes)**.** Let X be a set. An *equivalence relation on* X is a subset $R \subseteq X \times X$ satisfying the following properties for all $x, y, z \in X$:

Reflexivity: $(x, x) \in R$;

Symmetry: $(x, y) \in R$ if and only if $(y, x) \in R$; and

Transitivity: if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$.

If R is an equivalence relation, we often write $x \sim_R y$, or simply $x \sim y$, to mean $(x, y) \in R$. For convenience we may refer to the equivalence relation by the symbol $\sim$, saying that $\sim$ is an equivalence relation on X.

An *equivalence class of* $\sim$ is a subset $A \subseteq X$ such that

- A is nonempty, $A \neq \varnothing$;
- if $x \in A$ and $x' \in A$, then $x \sim x'$; and
- if $x \in A$ and $x \sim y$, then $y \in A$.

Suppose that $\sim$ is an equivalence relation on X. The *quotient of* X *by* $\sim$, denoted $X/\sim$ is the set of equivalence classes of $\sim$.

Example 2.6.1.2. Let $\mathbb{Z}$ denote the set of integers. Define a relation $R \subseteq \mathbb{Z} \times \mathbb{Z}$ by

$$R = \{(x, y) \mid \exists n \in \mathbb{Z} \text{ such that } x + 7n = y\}.$$

Then R is an equivalence relation because $x + 7 * 0 = x$ (reflexivity); $x + 7 * n = y$ if and only if $y + 7 * (-n) = x$ (symmetry); and $x + 7n = y$ and $y + 7m = z$ together imply that $x + 7(m + n) = z$ (transitivity).

Exercise 2.6.1.3. Let X be the set of people on earth; define a binary relation $R \subseteq X \times X$ on X as follows. For a pair (x, y) of people, say $(x, y) \in R$ if x spends a lot of time thinking about y.

a.) Is this relation reflexive?

b.) Is it symmetric?

c.) Is it transitive?

◊

Example 2.6.1.4 (Partitions). An equivalence relation on a set X can be thought of as a way of partitioning X. A *partition of* X consists of a set I, called *the set of parts*, and for every element $i \in I$ a subset $X_i \subseteq X$ such that two properties hold:

- every element $x \in X$ is in some part (i.e. for all $x \in X$ there exists $i \in I$ such that $x \in X_i$); and
- no element can be found in two different parts (i.e. if $x \in X_i$ and $x \in X_j$ then $i = j$).

Given a partition of X, we define an equivalence relation $\sim$ on X by saying $x \sim x'$ if x and x' are in the same part (i.e. if there exists $i \in I$ such that $x, x' \in X_i$). The parts become the equivalence classes of this relation. Conversely, given an equivalence relation, one makes a partition on X by taking I to be the set of equivalence classes and for each $i \in I$ letting X_i be the elements in that equivalence class.

Exercise 2.6.1.5. Let X and B be sets and let $f\colon X \to B$ be a function. Define a subset $R \subseteq X \times X$ by

$$R = \{(x, y) \mid f(x) = f(y)\}.$$

a.) Is R an equivalence relation?

b.) Are all equivalence relations on X obtainable in this way (as the fibers of some function having domain X)?

c.) Does this viewpoint on equivalence classes relate to that of Example 2.6.1.4?

◊

Exercise 2.6.1.6. Take a set I of sets; i.e. suppose that for each element $i \in I$ you are given a set X_i. For every two elements $i, j \in I$ say that $i \sim j$ if X_i and X_j are isomorphic. Is this relation an equivalence relation on I? ◊

Lemma 2.6.1.7 (Generating equivalence relations)**.** *Let X be a set and $R \subseteq X \times X$ a subset. There exists a relation $S \subseteq X \times X$ such that*

- *S is an equivalence relation,*
- *$R \subseteq S$, and*
- *for any equivalence relation S' such that $R \subseteq S'$, we have $S \subseteq S'$.*

The relation S' will be called the equivalence relation generated by *R.*

Proof. Let L_R be the set of all equivalence relations on X that contain R; in other words, each element $\ell \in L_R$ is an equivalence relation, $\ell \in X \times X$. The set L_R is non-empty because $X \times X \subseteq X \times X$ is an equivalence relation. Let S denote the set of pairs $(x_1, x_2) \in X \times X$ that appear in every element of L_R. Note that $R \subseteq S$ by definition. We need only show that S is an equivalence relation.

It is clearly reflexive, because R is. If $(x, y) \in S$ then $(x, y) \in \ell$ for all $\ell \in L_R$. But since each ℓ is an equivalence relation, $(y, x) \in \ell$ too, so $(y, x) \in S$. This shows that S is symmetric. The proof that it is transitive is similar: if $(x, y) \in S$ and $(y, z) \in S$ then they are both in each ℓ which puts (x, z) in each ℓ, which puts it in S.

□

Remark 2.6.1.8. Let X be a set and $R \subseteq X \times X$ a relation. The proof of Lemma 2.6.1.7 has the benefit of working even if $|X| \geqslant \infty$, but it has the cost that it is not very intuitive, nor useful in practice when X is finite. The intuitive way to think about the idea of equivalence relation generated by R is as follows.

1. First add to R what is demanded by reflexivity, $R_1 := R \cup \{(x, x) \mid x \in X\}$.
2. Then add to R what is demanded by symmetry, $R_2 := R_1 \cup \{(x, y) \mid (y, x) \in R_1\}$.
3. Finally, add to R what is demanded by transitivity,

$$S = R_2 \cup \{(x, z) \mid (x, y) \in R_2, \text{ and } (y, z) \in R_2\}.$$

Exercise 2.6.1.9. Consider the set $\mathbb{R}$ of real numbers. Draw the coordinate plane $\mathbb{R} \times \mathbb{R}$, give it coordinates x and y. A binary relation on $\mathbb{R}$ is a subset $S \subseteq \mathbb{R} \times \mathbb{R}$, which can be drawn as a set of points in the plane.

a.) Draw the relation $\{(x, y) \mid y = x^2\}$.

b.) Draw the relation $\{(x, y) \mid y \geqslant x^2\}$.

c.) Let S_0 be the equivalence relation on $\mathbb{R}$ generated (in the sense of Lemma 2.6.1.7) by the empty set. Draw S as a subset of the plane.

d.) Consider the equivalence relation S_1 generated by $\{(1, 2), (1, 3)\}$. Draw S_1 in the plane. Highlight the equivalence class containing $(1, 2)$.

e.) The reflexivity property and the symmetry property have pleasing visualizations in $\mathbb{R} \times \mathbb{R}$; what are they?

f.) Is there a nice heuristic for visualizing the transitivity property?

◊

Exercise 2.6.1.10. Consider the binary relation $R = \{(n, n + 1) \mid n \in \mathbb{Z}\} \subseteq \mathbb{Z} \times \mathbb{Z}$.

a.) What is the equivalence relation generated by R?

b.) How many equivalence classes are there?

◊

Exercise 2.6.1.11. Suppose N is a network (or graph). Let X be the nodes of the network, and let $R \subseteq X \times X$ denote the relation such that $(x, y) \in R$ iff there exists an arrow connecting x to y. [14]

a.) What is the equivalence relation $\sim$ generated by R?

b.) What is the quotient $X/\sim$?

◊

2.6.2 Pushouts

Definition 2.6.2.1 (Pushout)**.** Suppose given the diagram of sets and functions below:

$$\begin{tikzcd}
W \arrow[r, "f"] \arrow[d, "g"'] & X \\
Y &
\end{tikzcd} \tag{2.35}$$

Its *fiber sum*, denoted $X \sqcup_W Y$, is defined as the quotient of $X \sqcup W \sqcup Y$ by the equivalence relation $\sim$ generated by $w \sim f(w)$ and $w \sim g(w)$ for all $w \in W$.

$$X \sqcup_W Y := (X \sqcup W \sqcup Y)/\sim \qquad \text{where } \forall w \in W, \quad w \sim f(w) \quad \text{and} \quad w \sim g(w).$$

[14] The word *iff* means "if and only if". In this case we are saying that the pair (x, y) is in R if and only if there exists an arrow connecting x and y.

There are obvious inclusions $i_1\colon X \to X \sqcup_W Y$ and $i_2\colon Y \to X \sqcup_W Y$. [15] Note that if $Z = X \sqcup_W Y$ then the diagram

$$\begin{tikzcd}
W \arrow[r, "g"] \arrow[d, "f"'] & Y \arrow[d, "i_2"] \\
X \arrow[r, "i_1"'] & Z \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{2.36}$$

commutes. Given the setup of Diagram 2.35 we define the *pushout of X and Y over W* to be any set Z for which we have an isomorphism $Z \xrightarrow{\cong} X \sqcup_W Y$. The corner symbol $\ulcorner$ in Diagram 2.36 indicates that Z is the pushout.

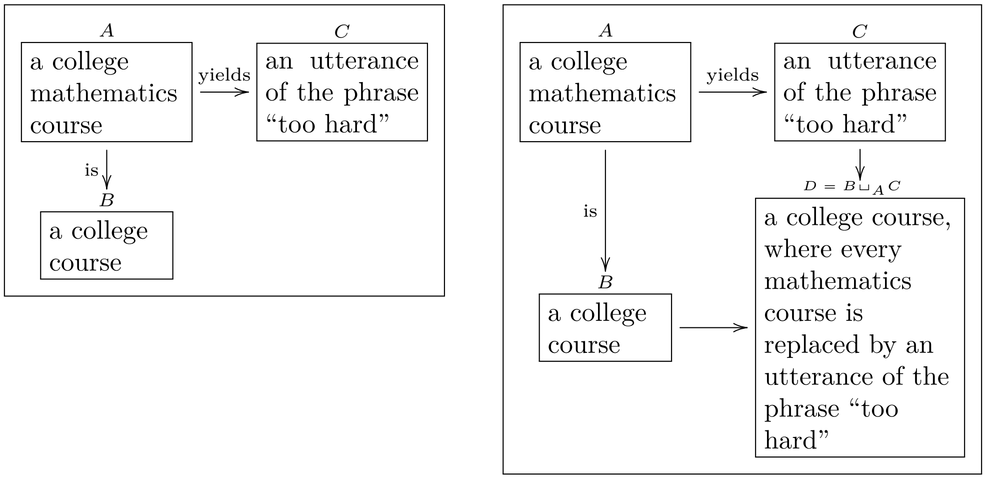
Example 2.6.2.2. Let $X = \{x \in \mathbb{R} \mid 0 \leqslant x \leqslant 1\}$ be the set of numbers between 0 and 1, inclusive, let $Y = \{y \in \mathbb{R} \mid 1 \leqslant y \leqslant 2\}$ by the set of numbers between 1 and 2, inclusive, and let $W = \{1\}$. Then the pushout $X \xleftarrow{f} W \xrightarrow{g} Y$, where f and g are the "obvious" functions $(1 \mapsto 1)$ is $X \sqcup_W Y \cong \{z \in \mathbb{R} \mid 0 \leqslant z \leqslant 2\}$, as expected. When we eventually get to general colimits, one can check that the whole real line can be made by patching together intervals in this way.

Example 2.6.2.3 (Pushout). In each example below, the diagram to the right is intended to be a pushout of the diagram to the left. The new object, D, is the union of B and C, but instances of A are equated to their B and C aspects. This will be discussed after the two diagrams.

$$\boxed{\begin{tikzcd}
\overset{A}{\boxed{\text{a cell in the shoulder}}} \arrow[r, "\text{is}"] \arrow[d, "\text{is}"'] & \overset{C}{\boxed{\text{a cell in the arm}}} \\
\overset{B}{\boxed{\text{a cell in the torso}}} &
\end{tikzcd}}
\quad
\boxed{\begin{tikzcd}
\overset{A}{\boxed{\text{a cell in the shoulder}}} \arrow[r, "\text{is}"] \arrow[d, "\text{is}"'] & \overset{C}{\boxed{\text{a cell in the arm}}} \arrow[d] \\
\overset{B}{\boxed{\text{a cell in the torso}}} \arrow[r] & \overset{D=B\sqcup_A C}{\boxed{\text{a cell in the torso or arm}}}
\end{tikzcd}} \tag{2.37}$$

In the left-hand olog (2.37, the two arrows are inclusions: the author considers every cell in the shoulder to be both in the arm and in the torso. The pushout is then just the union, where cells in the shoulder are not double-counted.

[15] Note that our term inclusions is not too good, because it seems to suggest that i_1 and i_2 are injective (see Definition 2.7.5.1) and this is not always the case.

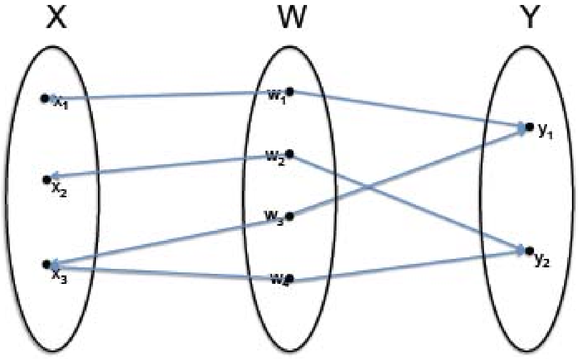

A
a college mathematics course
yields
C
an utterance of the phrase “too hard”
is
B
a college course
A
a college mathematics course
yields
C
an utterance of the phrase “too hard”
is
$D = B \sqcup_A C$
B
a college course
a college course, where every mathematics course is replaced by an utterance of the phrase “too hard”

(2.38)

In Olog (2.37), the shoulder is seen as part of the arm and part of the torso. When taking the union of these two parts, we do not want to “double-count” the shoulder (as would be done in the coproduct $B \sqcup C$, see Example 2.4.2.12). Thus we create a new type A for cells in the shoulder, which are considered the same whether viewed as cells in the arm or cells in the torso. In general, if one wishes to take two things and glue them together, with A as the glue and with B and C as the two things to be glued, the union is the pushout $B \sqcup_A C$. (A nice image of this can be seen in the setting of topological spaces, see Example 4.5.3.30.)

In Olog (2.38), if every mathematics course is simply “too hard,” then when reading off a list of courses, each math course will not be read aloud but simply read as “too hard.” To form D we begin by taking the union of B and C, and then we consider everything in A to be the same whether one looks at it as a course or as the phrase “too hard.” The math courses are all blurred together as one thing. Thus we see that the power to equate different things can be exercised with pushouts.

Exercise 2.6.2.4. Let W, X, Y be as drawn and $f\colon W \to X$ and $g\colon W \to Y$ the indicated functions.



The pushout of the diagram $X \xleftarrow{f} W \xrightarrow{g} Y$ is a set P. Write down the cardinality of $P \cong \underline{n}$ as a natural number $n \in \mathbb{N}$. ◊

Exercise 2.6.2.5. Suppose that $W = \varnothing$; what can you say about $X \sqcup_W Z$? ◊

Exercise 2.6.2.6. Let $W := \mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of natural numbers, let $X = \mathbb{Z}$ denote the set of integers, and let $Y = \{☺\}$ denote a one-element set. Define $f\colon W \to X$ by $f(w) = -(w+1)$, and define $g\colon W \to Y$ to be the unique map. Describe the set $X \sqcup_W Y$. ◊

Exercise 2.6.2.7. Let $i\colon R \subseteq X \times X$ be an equivalence relation (see Example 2.1.2.3 for notation). Composing with the projections $\pi_1, \pi_2\colon X \times X \to X$, we have two maps $\pi_1 \circ i, \colon R \to X$ and $\pi_2 \circ i\colon R \to X$.

a.) What is the pushout

$$X \xleftarrow{\pi_1 \circ i} R \xrightarrow{\pi_2 \circ i} X?$$

b.) If $i\colon R \subseteq X \times X$ is not assumed to be an equivalence relation, we can still define the pushout above. Is there a relationship between the pushout $X \xleftarrow{\pi_1 \circ i} R \xrightarrow{\pi_2 \circ i} X$ and the equivalence relation generated by $R \subseteq X \times X$?

◊

Lemma 2.6.2.8 (Universal property for pushout)**.** *Suppose given the diagram of sets and functions as below.*

$$\begin{tikzcd}
W \arrow[r, "u"] \arrow[d, "t"'] & Y \\
X &
\end{tikzcd}$$

For any set A and commutative solid arrow diagram as below (i.e. functions $f\colon X \to A$ and $g\colon Y \to A$ such that $f \circ t = g \circ u$),

$$\begin{tikzcd}
& W \arrow[dl, "t"'] \arrow[dr, "u"] & \\
X \arrow[r, "f"] \arrow[ddr, bend right, "i_1"'] & A & Y \arrow[l, "g"'] \arrow[ddl, bend left, "i_2"] \\
& & \\
& X \sqcup_W Y \arrow[uu, dashed, "\exists!"] &
\end{tikzcd}$$

(2.39)

there exists a unique arrow $\begin{cases} f \\ g \end{cases} \colon X \sqcup_W Y \to A$ *making everything commute,*

$$f = \begin{cases} f \\ g \end{cases} \circ i_1 \qquad \textit{and} \qquad g = \begin{cases} f \\ g \end{cases} \circ i_2.$$

2.6.3 Other finite colimits

Definition 2.6.3.1. [Coequalizer]

Suppose given two parallel arrows

$$\begin{tikzcd} X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y. \end{tikzcd} \qquad\qquad \begin{tikzcd} X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \arrow[r, "q"] & Coeq(f,g) \end{tikzcd} \tag{2.40}$$

The *coequalizer of f and g* is the commutative diagram as to the right in (2.40), where we define

$$Coeq(f,g) := Y \;/\; f(x) \sim g(x)$$

i.e. the coequalizer of f and g is the quotient of Y by the equivalence relation generated by $\{(f(x), g(x)) \mid x \in X\} \subseteq Y \times Y$

Exercise 2.6.3.2. Let $X = \mathbb{R}$ be the set of real numbers. What is the coequalizer of the two maps $X \to X$ given by $x \mapsto x$ and $x \mapsto (x+1)$ respectively? $\lozenge$

Exercise 2.6.3.3. Find a universal property enjoyed by the coequalizer of two arrows. $\lozenge$

Exercise 2.6.3.4 (Initial object). An initial set is a set S such that for every set A, there exists a unique function $S \to A$.

a.) Find an initial set.

b.) Do you think that the notion *initial set* belongs in this section (Section 2.6)? How so? If coproducts, pushouts, and coequalizers are all colimits, what do colimits have in common?

$\lozenge$

2.7 Other notions in Set

In this section we discuss some left-over notions in the category of Sets.

2.7.1 Retractions

Definition 2.7.1.1. Suppose we have a function $f\colon X \to Y$ and a function $g\colon Y \to X$ such that $g \circ f = \mathrm{id}_X$. In this case we call f a *retract section* and we call g a *retract projection.*

Exercise 2.7.1.2. Create an olog that includes sets X and Y, and functions $f\colon X \to Y$ and $g\colon Y \to X$ such that $g \circ f = \mathrm{id}_X$ but such that $f \circ g \neq \mathrm{id}_Y$; that is, such that f is a retract section but not an isomorphism. $\lozenge$

2.7.2 Currying

Currying is the idea that when a function takes many inputs, we can input them one at a time or all at once. For example, consider the function that takes a material M and an extension E and returns the force transmitted through the material when it is pulled to that extension. This is a function $e\colon \ulcorner\text{a material}\urcorner \times \ulcorner\text{an extension}\urcorner \to \ulcorner\text{a force}\urcorner$. This function takes two inputs at once, but it is convenient to "curry" the second input. Recall that $\mathrm{Hom}_{\mathbf{Set}}(\ulcorner\text{an extension}\urcorner, \ulcorner\text{a force}\urcorner)$ is the set of theoretical force-extension curves. Currying transforms e into a function

$$e' \colon \ulcorner\text{a material}\urcorner \to \mathrm{Hom}_{\mathbf{Set}}(\ulcorner\text{an extension}\urcorner, \ulcorner\text{a force}\urcorner).$$

This is a more convenient way to package the same information.

In fact, it may be convenient to repackage this information another way. For any extension, we may want the function that takes a material and returns how much force it can transmit at that extension. This is a function

$$e'' \colon \ulcorner\text{an extension}\urcorner \to \mathrm{Hom}_{\mathbf{Set}}(\ulcorner\text{a material}\urcorner, \ulcorner\text{a force}\urcorner).$$

Notation 2.7.2.1. Let A and B be sets. We sometimes denote the set of functions from A to B by

$$B^A := \mathrm{Hom}_{\mathbf{Set}}(A, B). \tag{2.41}$$

Exercise 2.7.2.2. For a finite set A, let $|A| \in \mathbb{N}$ denote the cardinality of (number of elements in) A. If A and B are both finite (including the possibility that one or both are empty), is it always true that $|B^A| = |B|^{|A|}$? ◊

Proposition 2.7.2.3 (Currying)**.** *Let A denote a set. For any sets X, Y there is a bijection*

$$\phi \colon \mathrm{Hom}_{\mathbf{Set}}(X \times A, Y) \xrightarrow{\cong} \mathrm{Hom}_{\mathbf{Set}}(X, Y^A). \tag{2.42}$$

Proof. Suppose given $f \colon X \times A \to Y$. Define $\phi(f) \colon X \to Y^A$ as follows: for any $x \in X$ let $\phi(f)(x) \colon A \to Y$ be defined as follows: for any $a \in A$, let $\phi(f)(x)(a) := f(x, a)$.

We now construct the inverse, $\psi \colon \mathrm{Hom}_{\mathbf{Set}}(X, Y^A) \to \mathrm{Hom}_{\mathbf{Set}}(X \times A, Y)$. Suppose given $g \colon X \to Y^A$. Define $\psi(g) \colon X \times A \to Y$ as follows: for any pair $(x, a) \in X \times A$ let $\psi(g)(x, a) := g(x)(a)$.

Then for any $f \in \mathrm{Hom}_{\mathbf{Set}}(X \times A, Y)$ we have $\psi \circ \phi(f)(x, a) = \phi(f)(x)(a) = f(x, a)$, and for any $g \in \mathrm{Hom}_{\mathbf{Set}}(X, Y^A)$ we have $\phi \circ \psi(g)(x)(a) = \psi(g)(x, a) = g(x)(a)$, Thus we see that ϕ is an isomorphism as desired.

□

Exercise 2.7.2.4. Let $X = \{1, 2\}, A = \{a, b\}$, and $Y = \{x, y\}$.

a.) Write down three distinct elements of $L := \mathrm{Hom}_{\mathbf{Set}}(X \times A, Y)$.

b.) Write down all the elements of $M := \mathrm{Hom}_{\mathbf{Set}}(A, Y)$.

c.) For each of the three elements $\ell \in L$ you chose in part (a), write down the corresponding function $\phi(\ell) \colon X \to M$ guaranteed by Proposition 2.7.2.3.

◊

Exercise 2.7.2.5. Let A and B be sets. We know that $\mathrm{Hom}_{\mathbf{Set}}(A, B) = B^A$, so we have a function $\mathrm{id}_{B^A} \colon \mathrm{Hom}_{\mathbf{Set}}(A, B) \to B^A$. Look at Proposition 2.7.2.3, making the substitutions $X = \mathrm{Hom}_{\mathbf{Set}}(A, B)$, $Y = B$, and $A = A$. Consider the function

$$\phi^{-1} \colon \mathrm{Hom}_{\mathbf{Set}}(\mathrm{Hom}_{\mathbf{Set}}(A, B), B^A) \to \mathrm{Hom}_{\mathbf{Set}}(\mathrm{Hom}_{\mathbf{Set}}(A, B) \times A, B)$$

obtained as the inverse of (2.42). We have a canonical element id_{B^A} in the domain of ϕ^{-1}. We can apply the function ϕ^{-1} and obtain an element $ev = \phi^{-1}(\mathrm{id}_{B^A}) \in \mathrm{Hom}_{\mathbf{Set}}(\mathrm{Hom}_{\mathbf{Set}}(A, B) \times A, B)$, which is itself a function,

$$ev \colon \mathrm{Hom}_{\mathbf{Set}}(A, B) \times A \to B.$$

a.) Describe the function ev in terms of how it operates on elements in its domain.

b.) Why might one be tempted to denote this function by ev?

$\Diamond$

If $n \in \mathbb{N}$ is a natural number, recall from (2.6) that there is a nice set $\underline{n} = \{1, 2, \ldots, n\}$. If A is a set, we often make the abbreviation

$$A^n := A^{\underline{n}}. \tag{2.43}$$

Exercise 2.7.2.6. In Example 2.4.1.7 we said that $\mathbb{R}^2$ is an abbreviation for $\mathbb{R} \times \mathbb{R}$, but in (2.43) we say that $\mathbb{R}^2$ is an abbreviation for $\mathbb{R}^{\underline{2}}$. Use Exercise 2.1.2.14, Proposition 2.7.2.3, Exercise 2.4.2.10, and the fact that 1+1=2, to prove that these are isomorphic, $\mathbb{R}^{\underline{2}} \cong \mathbb{R} \times \mathbb{R}$.

(The answer to Exercise 2.1.2.14 was $A = \{☺\}$: i.e. $\mathrm{Hom}_{\mathbf{Set}}(\{☺\}, X) \cong X$ for all X.)

$\Diamond$

2.7.3 Arithmetic of sets

Proposition 2.7.3.1 summarizes the properties of products, coproducts, and exponentials, and shows them all in a familiar light, namely that of arithmetic. In fact, one can think of the natural numbers as literally being the isomorphism classes of finite sets—that's what they are used for in counting. Consider the standard procedure for counting the elements of a set S, say cows in a field: one points to an element in S and simultaneously says "1", points to another element in S and simultaneously says "2", and so on until finished. This procedure amounts to nothing more than creating an isomorphism (one-to-one mapping) between S and some set $\underline{n}$.

Again, the natural numbers are the isomorphism classes of finite sets. Their behavior, i.e. the arithmetic of natural numbers, reflects the behavior of sets. For example the fact that multiplication distributes over addition is a fact about grids of dots as in Example 2.4.1.2. The following proposition lays out such arithmetic properties of sets.

In this proposition, we denote the coproduct of two sets A and B by the notation $A + B$ rather than $A \sqcup B$. It is a reasonable notation in general, and one that is often used.

Proposition 2.7.3.1. *The following isomorphisms exist for any sets A, B, and C (except for one caveat, see Exercise 2.7.3.2).*

- $A + \underline{0} \cong A$
- $A + B \cong B + A$
- $(A + B) + C \cong A + (B + C)$
- $A \times \underline{0} \cong \underline{0}$
- $A \times \underline{1} \cong A$
- $A \times B \cong B \times A$
- $(A \times B) \times C \cong A \times (B \times C)$
- $A \times (B + C) \cong (A \times B) + (A \times C)$

- $A^{\underline{0}} \cong \underline{1}$
- $A^{\underline{1}} \cong A$
- $\underline{0}^A \cong \underline{0}$
- $\underline{1}^A \cong \underline{1}$
- $A^{B+C} \cong A^B \times A^C$
- $(A^B)^C \cong A^{B \times C}$

Exercise 2.7.3.2. Everything in Proposition 2.7.3.1 is true except in one case, namely that of

$$\underline{0}^{\underline{0}}.$$

In this case, we get conflicting answers, because for any set A, including $A = \varnothing = \underline{0}$, we have claimed both that $A^{\underline{0}} \cong \underline{1}$ and that $\underline{0}^A \cong \underline{0}$.

What is the correct answer for $\underline{0}^{\underline{0}}$, based on the definitions of $\underline{0}$ and $\underline{1}$, given in (2.6), and of A^B, given in (2.41)? ◊

Exercise 2.7.3.3. It is also true of natural numbers that if $a, b \in \mathbb{N}$ and $ab = 0$ then either $a = 0$ or $b = 0$. Is the analogous statement true of all sets? ◊

Proposition 2.7.3.1 is in some sense about isomorphisms. It says that understanding isomorphisms of sets reduces to understanding natural numbers. But note that there is much more going on in **Set** than isomorphisms; in particular there are functions that are not invertible.

In grade school you probably never saw anything that looked like this:

$$5^3 \times 3 \longrightarrow 5$$

And yet in Exercise 2.7.2.5 we found a function $ev\colon B^A \times A \to B$ that exists for any sets A, B. This function ev is not an isomorphism so it somehow does not show up as an equation of natural numbers. But it still has important meaning. [16] In terms of mere number, it looks like we are being told of an important function $\underline{575} \to \underline{5}$, which is bizarre. The issue here is precisely the one you confronted in Exercise 2.1.2.13.

Exercise 2.7.3.4. Explain why there is a canonical function $\underline{5}^{\underline{3}} \times \underline{3} \longrightarrow \underline{5}$ but not a canonical function $\underline{575} \to \underline{5}$. ◊

Slogan 2.7.3.5.

> “ *It is true that a set is isomorphic to any other set with the same number of elements, but don't be fooled into thinking that the study of sets reduces to the study of numbers. Functions that are not isomorphisms cannot be captured within the framework of numbers.* ”

[16] Roughly, the existence of $ev\colon \underline{5}^{\underline{3}} \times \underline{3} \longrightarrow \underline{5}$ says that given a dot in a $5 \times 5 \times 5$ grid of dots, and given one of the three axes, you can tell me the coordinate of that dot along that axis.

2.7.4 Subobjects and characteristic functions

Definition 2.7.4.1. For any set B, define the *power set of* B, denoted $\mathbb{P}(B)$, to be the set of subsets of B.

Exercise 2.7.4.2.

a.) How many elements does $\mathbb{P}(\varnothing)$ have?

b.) How many elements does $\mathbb{P}(\{☺\})$ have?

c.) How many elements does $\mathbb{P}(\{1, 2, 3, 4, 5, 6\})$ have?

d.) Any idea why they may have named it "power set"?

◊

2.7.4.3 Simplicial complexes

Definition 2.7.4.4. Let V be a set and let $\mathbb{P}(V)$ be its powerset. A subset $X \subseteq \mathbb{P}(V)$ is called *downward-closed* if, for every $u \in X$ and every $u' \subseteq u$, we have $u' \in X$. We say that X *contains all atoms* if for every $v \in V$ the singleton set $\{v\}$ is an element of X.

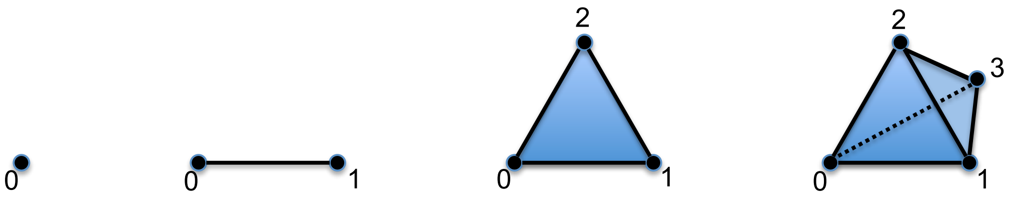
A *simplicial complex* is a pair (V, X) where V is a set and $X \subseteq \mathbb{P}(V)$ is a downward-closed subset that contains all atoms. The elements of X are called *simplices* (singular: *simplex*). Any subset $u \subseteq V$ has a cardinality $|u|$, so we have a function $X \to \mathbb{N}$ sending each simplex to its cardinality. The set of simplices with cardinality $n + 1$ is denoted X_n and each element $x \in X_n$ is called an *n-simplex.* [17] Since X contains all atoms (subsets of cardinality 1), we have $X_0 \cong V$, and we may also call the 0-simplices *vertices.* We sometimes call the 1-simplices *edges.* [18]

Since $X_0 \cong V$, we may denote a simplicial complex (V, X) simply by X.

Example 2.7.4.5. Let $n \in \mathbb{N}$ be a natural number and let $V = \underline{n+1}$. Define *the n-simplex*, denoted Δ^n, to be the simplicial complex $\mathbb{P}(V) \subseteq \mathbb{P}(V)$, i.e. the whole power set, which indeed is downward-closed and contains all atoms.

We can draw a simplicial complex X by first putting all the vertices on the page as dots. Then for every $x \in X_1$, we see that $x = \{v, v'\}$ consists of 2 vertices, so we draw an edge connecting v and v'. For every $y \in X_2$ we see that $y = \{w, w', w''\}$ consists of 3 vertices, so we draw a (filled-in) triangle connecting them. All three edges will be drawn too because X is assumed to be downward closed.

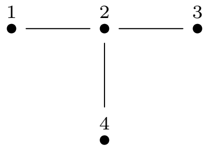
Thus, the 0-simplex Δ^0, the 1-simplex Δ^1, the 2-simplex Δ^2, and the 3-simplex Δ^3 are drawn here:



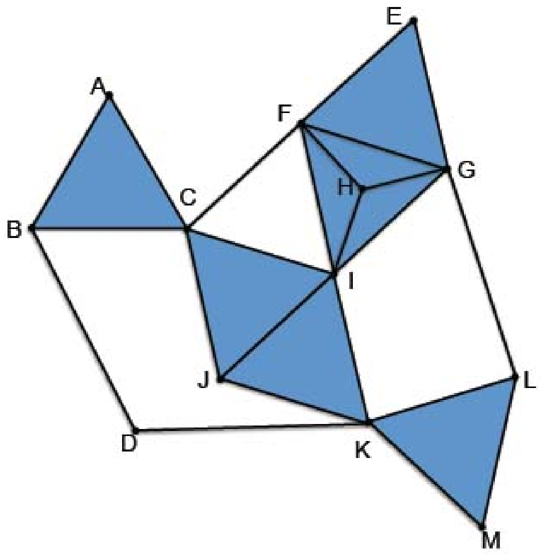
[17] It is annoying at first that the set of subsets with cardinality 1 is denoted X_0, etc. But this is standard convention because as we will see, X_n will be n-dimensional.

[18] The reason we wrote $X_0 \cong V$ rather than $X_0 = V$ is that X_0 is the set of 1-element subsets of V. So if $V = \{a, b, c\}$ then $X_0 = \{\{a\}, \{b\}, \{c\}\}$. This is really just pedantry.

The n-simplices for various n's are in no way all of the simplicial complexes. In general a simplicial complex is a union or "gluing together" of simplices in a prescribed manner. For example, consider the simplicial complex X with vertices $X_0 = \{1, 2, 3, 4\}$, edges $X_1 = \{\{1, 2\}, \{2, 3\}, \{2, 4\}\}$, and no higher simplices $X_2 = X_3 = \cdots = \varnothing$. We might draw X as follows:



Exercise 2.7.4.6. Let X be the following simplicial complex, so that $X_0 = \{A, B, \ldots, M\}$.



In this case X_1 consists of elements like $\{A, B\}$ and $\{D, K\}$ but not $\{D, J\}$.

Write out X_2 and X_3 (hint: the drawing of X indicates that X_3 should have one element). ◊

Exercise 2.7.4.7. The 2-simplex Δ^2 is drawn as a filled-in triangle with vertices $V = \{1, 2, 3\}$. There is a simplicial complex $X = \partial\Delta^2$ that would be drawn as an empty triangle with the same set of vertices.

a.) Draw Δ^2 and X side by side and make clear the difference.

b.) Write down the data for X as a simplicial complex. In other words what are the sets $X_0, X_1, X_2, X_3, \ldots$?

◊

2.7.4.8 Subobject classifier

Definition 2.7.4.9. Define the *subobject classifier* for **Set**, denoted Ω, to be the set $\Omega := \{True, False\}$, together with the function $\{☺\} \to \Omega$ sending the unique element to $True$.

Proposition 2.7.4.10. *Let B be a set. There is an isomorphism*

$$\phi\colon \mathrm{Hom}_{\mathbf{Set}}(B, \Omega) \xrightarrow{\cong} \mathbb{P}(B).$$

Proof. Given a function $f\colon B \to \Omega$, let $\phi(f) = \{b \in B \mid f(b) = \mathit{True}\} \subseteq B$. We now construct a function $\psi\colon \mathbb{P}(B) \to \mathrm{Hom}_{\mathbf{Set}}(B, \Omega)$ to serve as the inverse of ϕ. Given a subset $B' \subseteq B$, define $\psi(B')\colon B \to \Omega$ as follows:

$$\psi(i)(b) = \begin{cases} \mathit{True} & \text{if } b \in B', \\ \mathit{False} & \text{if } b \notin B'. \end{cases}$$

One checks easily that ϕ and ψ are mutually inverse.

□

Definition 2.7.4.11 (Characteristic function)**.** Given a subset $B' \subseteq B$, we call the corresponding function $B \to \Omega$ the *characteristic function of B' in B.*

Let B be any set and let $\mathbb{P}(B)$ be its power set. By Proposition 2.7.4.10 there is a bijection between $\mathbb{P}(B)$ and Ω^B. Since Ω has cardinality 2, the cardinality of $\mathbb{P}(B)$ is $2^{|B|}$, which explains the correct answer to Exercise 2.7.4.2.

Exercise 2.7.4.12. Let $f\colon A \to \Omega$ denote the characteristic function of some $A' \subseteq A$, and define $A'' \subseteq A$ to be its complement, $A'' := A - A'$ (i.e. $a \in A''$ if and only if $a \notin A'$).

a.) What is the characteristic function of $A'' \subseteq A$?

b.) Can you phrase it in terms of some function $\Omega \to \Omega$?

◊

2.7.5 Surjections, injections

The classical definition of injections and surjections involves elements, which we give now. But a more robust notion involves all maps and will be given in Proposition 2.7.5.4.

Definition 2.7.5.1. Let $f\colon X \to Y$ be a function. We say that f is *surjective* if, for all $y \in Y$ there exists some $x \in X$ such that $f(x) = y$. We say that f is *injective* if, for all $x \in X$ and all $x' \in X$ with $f(x) = f(x')$ we have $x = x'$.

A function that is both injective and surjective is called *bijective.*

Remark 2.7.5.2. It turns out that a function that is bijective is always an isomorphism and that all isomorphisms are bijective. We will not show that here, but it is not too hard; see for example [Big, Theorem 5.4].

Definition 2.7.5.3 (Monomorphisms, epimorphisms)**.** Let $f\colon X \to Y$ be a function.

We say that f is a *monomorphism* if for all sets A and pairs of functions $g, g'\colon A \to X$,

$$\begin{tikzcd} A \arrow[r, bend left, "g"] \arrow[r, bend right, "g'"'] & X \arrow[r, "f"] & Y \end{tikzcd}$$

if $f \circ g = f \circ g'$ then $g = g'$.

We say that f is an *epimorphism* if for all sets B and pairs of functions $h, h' \colon Y \to B$,

$$\begin{tikzcd}
X \arrow[r, "f"] & Y \arrow[r, bend left, "h"] \arrow[r, bend right, "h'"'] & B
\end{tikzcd}$$

if $h \circ f = h' \circ f$ then $h = h'$.

Proposition 2.7.5.4. *Let $f \colon X \to Y$ be a function. Then f is injective if and only if it is a monomorphism; f is surjective if and only if it is an epimorphism.*

Proof. If f is a monomorphism it is clearly injective by putting $A = \{☺\}$. Suppose that f is injective and let $g, g' \colon A \to X$ be functions such that $f \circ g = f \circ g'$, but suppose for contradiction that $g \neq g'$. Then there is some element $a \in A$ such $g(a) \neq g'(a) \in X$. But by injectivity $f(g(a)) \neq f(g'(a))$, contradicting $f \circ g = f \circ g'$.

Suppose that $f \colon X \to Y$ is an epimorphism and choose some $y_0 \in Y$ (noting that if Y is empty then the claim is vacuously true). Let $h \colon Y \to \Omega$ denote the characteristic function of the subset $\{y_0\} \subseteq Y$ and let $h' \colon Y \to \Omega$ denote the characteristic function of $\varnothing \subseteq Y$; note that $h(y) = h'(y)$ for all $y \neq y_0$. Then since f is an epimorphism and $h \neq h'$, we must have $h \circ f \neq h' \circ f$, so there exists $x \in X$ with $h(f(x)) \neq h'(f(x))$, which implies that $f(x) = y_0$. This proves that f is surjective.

Finally, suppose that f is surjective, and let $h, h' \colon Y \to B$ be functions with $h \circ f = h' \circ f$. For any $y \in Y$, there exists some $x \in X$ with $f(x) = y$, so $h(y) = h(f(x)) = h'(f(x)) = h'(y)$. This proves that f is an epimorphism.

□

Proposition 2.7.5.5. *Let $f \colon X \to Y$ be a monomorphism. Then for any function $g \colon A \to Y$, the top map $f' \colon X \times_Y A \to A$ in the diagram*

$$\begin{tikzcd}
X \times_Y A \arrow[r, "f'"] \arrow[d, "g'"'] \arrow[dr, phantom, very near start, "\lrcorner"] & A \arrow[d, "g"] \\
X \arrow[r, "f"'] & Y
\end{tikzcd}$$

is a monomorphism.

Proof. To show that f' is a monomorphism, we take an arbitrary set B and two maps $m, n \colon B \to X \times_Y A$ such that $f' \circ m = f' \circ n$, denote that function by $p := f' \circ m \colon B \to A$. Now let $q = g' \circ m$ and $r = g' \circ n$. The diagram looks like this:

$$\begin{tikzcd}
B \arrow[r, shift left, "m"] \arrow[r, shift right, "n"'] \arrow[dr, shift left, "q"] \arrow[dr, shift right, "r"'] \arrow[rr, bend left, "p"] & X \times_Y A \arrow[r, "f'"] \arrow[d, "g'"'] \arrow[dr, phantom, very near start, "\lrcorner"] & A \arrow[d, "g"] \\
& X \arrow[r, "f"'] & Y
\end{tikzcd}$$

We have that

$$f \circ q = f \circ g' \circ m = g \circ f' \circ m = g \circ f' \circ n = f \circ g' \circ n = f \circ r$$

But we assumed that f is a monomorphism so this implies that $q = r$. By the universal property of pullbacks, Lemma 2.5.1.14, we have $m = n$.

□

Exercise 2.7.5.6. Show, in analogy to Proposition 2.7.5.5, that pushouts preserve epimorphisms. ◊

Example 2.7.5.7. Suppose an olog has a fiber product square

$$\begin{tikzcd}
X \times_Z Y \arrow[r, "g'"] \arrow[d, "f'"'] & Y \arrow[d, "f"] \\
X \arrow[r, "g"'] & Z
\end{tikzcd}$$

such that f is intended to be an injection and g is any map. [19] In this case, there are nice labeling systems for f', g', and $X \times_Z Y$. Namely:

- "is" is an appropriate label for f',
- the label for g is an appropriate label for g',
- (the label for X, then "which", then the label for g, then the label for Y) is an appropriate label for $X \times_Z Y$.

To give an explicit example,

$$\begin{tikzcd}
\overset{X \times_Z Y}{\boxed{\begin{array}{l}\text{a rib which is}\\ \text{made by a cow}\end{array}}} \arrow[r, "\text{is made by}"] \arrow[d, "\text{is}"'] & \overset{Y}{\boxed{\text{a cow}}} \arrow[d, "\text{is}"] \\
\overset{X}{\boxed{\text{a rib}}} \arrow[r, "\text{is made by}"'] & \overset{Z}{\boxed{\text{an animal}}}
\end{tikzcd}$$

Corollary 2.7.5.8. *Let $i \colon A \to X$ be a monomorphism. Then there is a fiber product square of the form*

$$\begin{tikzcd}
A \arrow[r, "f'"] \arrow[d, "i"'] \arrow[dr, phantom, "\lrcorner", very near start] & \{☺\} \arrow[d, "True"] \\
X \arrow[r, "f"'] & \Omega.
\end{tikzcd} \tag{2.44}$$

Proof. Let $X' \subseteq X$ denote the image of i and let $f \colon X \to \Omega$ denote the characteristic function of $X' \subseteq X$. Then it is easy to check that Diagram 2.44 is a pullback.

□

Exercise 2.7.5.9. Consider the subobject classifier Ω, the singleton $\{☺\}$ and the map $\{☺\} \xrightarrow{True} \Omega$ from Definition 2.7.4.9. Look at diagram 2.44 and in the spirit of Exercise 2.7.5.7, come up with a label for Ω, a label for $\{☺\}$, and a label for $True$. Given a label for X and a label for f, come up with a label for A, a label for i and a label for f', such that the English smoothly fits the mathematics. ◊

[19] Of course, this diagram is symmetrical, so the same ideas hold if g is an injection and f is any map.

2.7.6 Multisets, relative sets, and set-indexed sets

In this section we prepare ourselves for considering categories other than **Set**, by looking at some categories related to **Set**.

2.7.6.1 Multisets

Consider the set X of words in a given document. If $WC(X)$ is the wordcount of the document, we will not generally have $WC(X) = |X|$. The reason is that a set cannot contain the same element more than once, so words like "the" might be undercounted in $|X|$. A *multiset* is a set in which elements can be assigned a multiplicity, i.e. a number of times they are to be counted.

But if X and Y are multisets, what is the appropriate type of mapping from X to Y? Since every set is a multiset (in which each element has multiplicity 1), let's restrict ourselves to notions of mapping that agree with the usual one on sets. That is, if multisets X and Y happen to be sets then our mappings $X \to Y$ should just be functions.

Exercise 2.7.6.2.

a.) Come up with some notion of mapping for multisets that generalizes functions when the notion is restricted to sets.

b.) Suppose that $X = (1, 1, 2, 3)$ and $Y = (a, b, b, b)$, i.e. $X = \{1, 2, 3\}$ with 1 having multiplicity 2, and $Y = \{a, b\}$ with b having multiplicity 3. What are all the maps $X \to Y$ in your notion?

◊

In Chapter 4 we will be getting to the definition of category, and you can test whether your notion of mapping in fact defines a category. Here is my definition of mapping for multisets.

Definition 2.7.6.3. A *multiset* is a sequence $X := (E, B, \pi)$ where E and B are sets and $\pi\colon E \to B$ is a surjective function. We refer to E as the set of *element instances of* X, we refer to B as the set of *element names of* X, and we refer to π as the *naming function for* X. Given an element name $x \in B$, let $\pi^{-1}(x) \subseteq E$ be the preimage; the number of elements in $\pi^{-1}(x)$ is called the *multiplicity of* x.

Suppose that $X = (E, B, \pi)$ and $X' = (E', B', \pi')$ are multisets. A *mapping from* X *to* Y, denoted $f\colon X \to Y$, consists of a pair (f_1, f_0) such that $f_1\colon E \to E'$ and $f_0\colon B \to B'$ are functions and such that the following diagram commutes:

$$\begin{tikzcd}
E \arrow[r, "f_1"] \arrow[d, "\pi"'] & E' \arrow[d, "\pi'"] \\
B \arrow[r, "f_0"'] & B'.
\end{tikzcd} \tag{2.45}$$

Exercise 2.7.6.4. Suppose that a pseudo-multiset is defined to be almost the same as a multiset, except that π is not required to be surjective.

a.) Write down a pseudo-multiset that is not a multi-set.

b.) Describe the difference between the two notions in terms of multiplicities.

c.) Complexity of names aside, which do you think is a more useful notion: multiset or pseudo-multisets?

◊

Exercise 2.7.6.5. Consider the multisets described in Exercise 2.7.6.2.

a.) Write each of them in the form (E, B, π), as in Definition 2.7.6.3.

b.) In terms of the same definition, what are the mappings $X \to Y$?

c.) If we remove the restriction that diagram 2.45 must commute, how many mappings $X \to Y$ are there?

◊

2.7.6.6 Relative sets

Let's continue with our ideas from multisets, but now suppose that we have a fixed set B of names that we want to keep once and for all. Whenever someone discusses a set, each element must have a name in B. And whenever someone discusses a mapping, it must preserve the names. For example, if B is the set of English words, then every document consists of an ordered set mapping to B (e.g. $1 \mapsto \text{Suppose}, 2 \mapsto \text{that}, 3 \mapsto \text{we, etc.}$) A mapping from document A to document B would send each word found somewhere in A to the same word found somewhere in B. This notion is defined carefully below.

Definition 2.7.6.7 (Relative set)**.** Let B be a set. A *relative set over* B, or simply a *set over* B, is a pair (E, π) such that E is a set and $\pi\colon E \to B$ is a function. A *mapping of relative sets over* B, denoted $f\colon (E, \pi) \to (E', \pi')$, is a function $f\colon E \to E'$ such that the triangle below commutes, i.e. $\pi = \pi' \circ f$,

$$\begin{tikzcd} E \arrow[rr, "f"] \arrow[dr, "\pi"'] & & E' \arrow[dl, "\pi'"] \\ & B & \end{tikzcd}$$

Exercise 2.7.6.8. Given sets X, Y, Z and functions $f\colon X \to Y$ and $g\colon Y \to Z$, we can compose them to get a function $X \to Z$. If B is a set, if $(X, p), (Y, q)$, and (Z, r) are relative sets over B, and if $f\colon (X, p) \to (Y, q)$ and $g\colon (Y, q) \to (Z, r)$ are mappings, is there a reasonable notion of composition such that we get a mapping of relative sets $(X, p) \to (Z, r)$? Hint: draw diagrams. ◊

Exercise 2.7.6.9.

a.) Let $\{☺\}$ denote a set with one element. What is the difference between sets over $\{☺\}$ and simply sets?

b.) Describe the sets relative to $\varnothing$. How many are there?

◊

2.7.6.10 Indexed sets

Let A be a set. Suppose we want to assign to each element $a \in A$ a set S_a. This is called an A-indexed set. In category theory we are always interested in the legal mappings between two different structures of the same sort, so we need a notion of A-indexed mappings; we do the "obvious thing".

Example 2.7.6.11. Let C be a set of classrooms. For each $c \in C$ let P_c denote the set of people in classroom c, and let S_c denote the set of seats (chairs) in classroom c. Then P and S are C-indexed sets. The appropriate kind of mapping between them respects the indexes. That is, a mapping of multi-sets $P \to S$ should, for each classroom $c \in C$, be a function $P_c \to S_c$.[20]

Definition 2.7.6.12. Let A be a set. An *A-indexed set* is a collection of sets S_a, one for each element $a \in A$; for now we denote this by $(S_a)_{a\in A}$. If $(S'_a)_{a\in A}$ is another A-indexed set, a *mapping of A-indexed sets from $(S_a)_{a\in A}$ to $(S'_a)_{a\in A}$*, denoted

$$(f_a)_{a\in A} \colon (S_a)_{a\in A} \to (S'_a)_{a\in A}$$

is a collection of functions $f_a \colon S_a \to S'_a$, one for each element $a \in A$.

Exercise 2.7.6.13. Let $\{☺\}$ denote a one element set. What are $\{☺\}$-indexed sets and mappings between them? ◊

Exercise 2.7.6.14. There is a strong relationship between A-indexed sets and relative sets over A. What is it? ◊

[20] If we wanted to allow people from any classroom to choose a chair from just any classroom, category theory would tell us to reconsider P and S as sets, forgetting their indices. See Section 5.1.4.7.

Chapter 3

Categories and functors, without admitting it

In this chapter we begin to use our understanding of sets to build more interesting mathematical devices, each of which organizes our understanding of a certain kind of domain. For example, monoids organize our thoughts about agents acting on objects; groups are monoids except restricted to only allow agents to act reversibly. We will then study graphs, which are systems of nodes and arrows that can capture ideas like information flow through a network or model connections between building blocks in a material. We will discuss orders, which can be used to study taxonomies or hierarchies. Finally we take a mathematical look at databases, which actually subsume everything else in the chapter. Databases are connection patterns for structuring information.

We will see in Chapter 4 that everything we study in the present chapter is an example of a category. So is **Set**, the category of sets studied in Chapter 2. One way to think of a category is as a set of objects and a connection pattern between them; sets are objects (ovals full of dots if you wish) connected by functions. But each set is itself a category: the objects inside it are just disconnected! Just like a set has an interior view and an exterior view, so will all the categories in this chapter. Each monoid *is* a category, but there is also a category *of* monoids.

However, we will not really say the word "category" much if at all in this chapter. It seems preferable to let the ideas rise on their own accord as interesting structures in their own right before explaining that everything in site fits into a single framework. That will be the pleasant reward to come in Chapter 4.

3.1 Monoids

A common way to interpret phenomena we see around us is to say that agents are acting on objects. For example, in a computer drawing program, the user *acts on* the canvas in certain prescribed ways. Choices of actions from an available list can be performed in sequence to transform one image into another. As another example, one might investigate the notion that time *acts on* the position of hands on a clock in a prescribed way. A first rule for actions is this: the performance of a sequence of several actions is itself the performance of an action—a more complex action, but an action nonetheless.

Mathematical objects called *monoids* and *groups* are tasked with encoding the agent's perspective in all this, i.e. what the agent can do, and what happens when different actions are done in succession. A monoid can be construed as a set of actions, together with a formula that encodes how a sequence of actions is itself considered an action. A group is the same as a monoid, except that every action is required to be reversible. In this section we concentrate on monoids; we will get to groups in Section 3.2.

3.1.1 Definition and examples

Definition 3.1.1.1 (Monoid)**.** A *monoid* is a sequence $(M, e, \star)$, where M is a set, $e \in M$ is an element, and $\star\colon M \times M \to M$ is a function, such that the following conditions hold for all $m, n, p \in M$:

- $m \star e = m$,
- $e \star m = m$, and
- $(m \star n) \star p = m \star (n \star p)$.

We refer to e as the *identity element* and to $\star$ as the *multiplication formula* for the monoid. [1] We call the first two rules *identity laws* and the third rule the *associativity law* for monoids.

Remark 3.1.1.2. To be pedantic, the conditions from Definition 3.1.1.1 should be stated

- $\star(m, e) = m$,
- $\star(e, m) = m$, and
- $\star(\star(m, n), p) = \star(m, (\star(n, p))$.

The way they are written in Definition 3.1.1.1 is called *infix notation*, and we often use infix notation without mentioning it. That is, given a function $\cdot\colon A \times B \to C$, we may write $a \cdot b$ rather than $\cdot(a, b)$.

Example 3.1.1.3 (Additive monoid of natural numbers). Let $M = \mathbb{N}$ be the set of natural numbers. Let $e = 0$ and let $\star\colon M \times M \to M$ denote addition, so that $\star(4, 18) = 22$. Then the equations $m \star 0 = m$ and $0 \star m = m$ hold, and $(m \star n) \star p = m \star (n \star p)$. By assigning e and $\star$ in this way, we have "given $\mathbb{N}$ the structure of a monoid".

Remark 3.1.1.4. Sometimes we are working with a monoid $(M, e, \star)$, and the identity e and multiplication $\star$ are somehow clear from context. In this case we might refer to the set M as though it were the whole monoid. For example, if we were discussing the monoid from Example 3.1.1.3, we might refer to it as $\mathbb{N}$. The danger comes because sets may have multiple monoid structures, as we see below in Exercise 3.1.1.6.

Example 3.1.1.5 (Non-monoid). If M is a set, we might call a function $f\colon M \times M \to M$ an *operation on M*. For example, if $M = \mathbb{N}$ is the set of natural numbers, we can consider the operation $f\colon \mathbb{N} \to \mathbb{N}$ called exponentiation. For example $f(2, 5) = 2 * 2 * 2 * 2 * 2 = 32$ and $f(7, 2) = 49$. This is indeed an operation, but it is not part of any monoid. For one thing there is no possible unit. Trying the obvious choice of $e = 1$, we see that $a^1 = a$ (good), but that $1^a = 1$ (bad: we need it to be a). For another thing, this operation is not associative because in general $a^{b^c} \neq (a^b)^c$. For example, $2^{1^2} = 2$ but $(2^1)^2 = 4$.

[1] Although the function $\star\colon M \times M \to M$ is called the multiplication formula, it may have nothing to do with multiplication. It is nothing more than a formula for taking two inputs and returning an output; calling it "multiplication" is suggestive of its origins, rather than prescriptive of its behavior.

One might also attempt to consider an operation $f\colon M \times M \to M$ that, upon closer inspection, aren't even operations. For example, if $M = \mathbb{Z}$ then exponentiation is not even an operation. Indeed, $f(2, -1) = 2^{-1} = \frac{1}{2}$, and this is not an integer. To have a function $f\colon M \times M \to M$, we need that every element of the domain, in this case every pair of integers, has an output under f. So there is no such function f.

Exercise 3.1.1.6. Let $M = \mathbb{N}$ be the set of natural numbers. Taking $e = 1$, come up with a formula for $\star$ that gives $\mathbb{N}$ the structure of a monoid. ◊

Exercise 3.1.1.7. Come up with an operation on the set $M = \{1, 2, 3, 4\}$, i.e. a legitimate function $f\colon M \times M \to M$, such that f cannot be the multiplication formula for a monoid on M. That is, either it is not associative, or no element of M can serve as a unit. ◊

Exercise 3.1.1.8. In both Example 3.1.1.3 and Exercise 3.1.1.6, the monoids $(M, e, \star)$ satisfied an additional rule called *commutativity*, namely $m \star n = n \star m$ for every $m, n \in M$. There is a monoid $(M, e, \star)$ lurking in linear algebra textbooks that is not commutative; if you have background in linear algebra try to answer this: what M, e, and $\star$ might I be referring to? ◊

Exercise 3.1.1.9. Recall the notion of commutativity for monoids from Exercise 3.1.1.8.

a.) What is the smallest set M that you can give the structure of a non-commutative monoid?

b.) What is the smallest set M that you can give the structure of a monoid?

◊

Example 3.1.1.10 (Trivial monoid). There is a monoid with only one element, $M = (\{e\}, e, \star)$ where $\star\colon \{e\} \times \{e\} \to \{e\}$ is the unique function. We call this monoid *the trivial monoid*, and sometimes denote it $\underline{1}$.

Example 3.1.1.11. Suppose that $(M, e, \star)$ is a monoid. Given elements m_1, m_2, m_3, m_4 there are five different ways to parenthesize the product $m_1 \star m_2 \star m_3 \star m_4$, and the associativity law for monoids will show them all to be the same. We have

$$\begin{aligned}((m_1 \star m_2) \star m_3) \star m_4 &= (m_1 \star m_2) \star (m_3 \star m_4)\\ &= (m_1 \star (m_2 \star m_3)) \star m_4\\ &= m_1 \star (m_2 \star (m_3 \star m_4))\\ &= m_1 \star ((m_2 \star m_3) \star m_4)\end{aligned}$$

In fact, the product of any list of monoid elements is the same, regardless of parenthesization. Therefore, we can unambiguously write $m_1m_2m_3m_4m_5$ rather than any given parenthesization of it. This is known as the coherence theorem and can be found in [Mac].

3.1.1.12 Free monoids and finitely presented monoids

Definition 3.1.1.13. Let X be a set. A *list in* X is a pair (n, f) where $n \in \mathbb{N}$ is a natural number (called the *length of the list*) and $f\colon \underline{n} \to X$ is a function, where $\underline{n} = \{1, 2, \ldots, n\}$. We may denote such a list by

$$(n, f) = [f(1), f(2), \ldots, f(n)].$$

The *empty list* is the unique list in which $n = 0$; we may denote it by [ ]. Given an element $x \in X$ the *singleton list on* x is the list $[x]$. Given a list $L = (n, f)$ and a number $i \in \mathbb{N}$ with $i \leqslant n$, the *ith entry of* L is the element $f(i) \in X$.

Given two lists $L = (n, f)$ and $L' = (n', f')$, define the *concatenation of L and L'*, denoted $L +\!\!+ L'$, to be the list $(n + n', f +\!\!+ f')$, where $f +\!\!+ f' \colon \underline{n + n'} \to X$ is given on $i \leqslant n + n'$ by

$$(f +\!\!+ f')(i) := \begin{cases} f(i) & \text{if } i \leqslant n \\ f'(i-n) & \text{if } i \geqslant n+1 \end{cases}$$

Example 3.1.1.14. Let $X = \{a, b, c, \ldots, z\}$. The following are elements of $\mathrm{List}(X)$:

$$[a, b, c], \quad [p], \quad [p, a, a, a, p], \quad [\,], \quad \ldots$$

The concatenation of $[a, b, c]$ and $[p, a, a, a, p]$ is $[a, b, c, p, a, a, a, p]$. The concatenation of any list A with $[\,]$ is just A.

Definition 3.1.1.15. Let X be a set. The *free monoid generated by X* is the sequence $M := (\mathrm{List}(X), [\,], +\!\!+)$, where $\mathrm{List}(X)$ is the set of lists of elements in X, where $[\,] \in \mathrm{List}(X)$ is the empty list, and where $+\!\!+$ is the operation of list concatenation. We refer to X as the set of generators for the monoid M.

Exercise 3.1.1.16. Let $\{☺\}$ denote a one-element set.

a.) What is the free monoid generated by $\{☺\}$?

b.) What is the free monoid generated by $\varnothing$?

$\Diamond$

In the definition below, we will define a monoid M by specifying some generators and some relations. Lists of generators provide us all the possible ways to write elements of M. The relations allow us to have two such ways of writing the same element. The following definition is a bit dense, so see Example 3.1.1.19 for a concrete example.

Definition 3.1.1.17 (Presented monoid)**.** Let G be a finite set, let $n \in \mathbb{N}$ be a natural number, [2] and for each $1 \leqslant i \leqslant n$, let m_i and m'_i be elements of $\mathrm{List}(G)$. [3] The *monoid presented by generators G and relations* $\{(m_i, m'_i) \mid 1 \leqslant i \leqslant n\}$ is the monoid $\mathcal{M} = (M, e, \star)$ defined as follows. Let $\sim$ denote the equivalence relation on $\mathrm{List}(G)$ generated by $\{(xm_iy \sim xm'_iy) \mid x, y \in \mathrm{List}(G), 1 \leqslant i \leqslant n\}$, and define $M = \mathrm{List}(G)/\sim$. Let $e = [\,]$ and let $a * b$ be obtained by concatenating representing lists.

Remark 3.1.1.18. Every free monoid is a presented monoid, because we can just take the set of relations to be empty.

Example 3.1.1.19. Let $G = \{a, b, c, d\}$. Think of these as buttons that can be pressed. The free monoid $\mathrm{List}(G)$ is the set of all ways of pressing buttons, e.g. pressing a then a then c then c then d corresponds to the list $[a, a, c, c, d]$. The idea of presented monoids is that you notice that pressing $[a, a, c]$ always gives the same result as pressing $[d, d]$. You also notice that pressing $[c, a, c, a]$ is the same thing as doing nothing.

In this case, we would have $m_1 = [a, a, c]$, $m'_1 = [d, d]$, and $m_2 = [c, a, c, a]$, $m'_2 = [\,]$ and relations $\{(m_1, m'_1), (m_2, m'_2)\}$. Really this means that we're equating m_1 with m'_1 and m_2 with m'_2, which for convenience we'll write out:

$$[a, a, c] = [d, d] \qquad \text{and} \qquad [a, c, a, c] = [\,]$$

[2] The number $n \in \mathbb{N}$ is going to stand for the number of relations we declare.

[3] Each m_i and m'_i are going to be made equal in the set M.

To see how this plays out, we give an example of a calculation in $M = \text{List}(G)/\sim$. Namely,

$$\begin{aligned}[b,c,b,d,d,a,c,a,a,c,d] = [b,c,b,a,a,c,a,c,a,a,c,d] &= [b,c,b,a,a,a,c,d] \\ &= [b,c,b,a,d,d,d].\end{aligned}$$

Application 3.1.1.20 (Buffer). Let $G = \{a,b,c,\dots z\}$. Suppose we have a buffer of 32 characters and we want to consider the set of lists of length at most 32 to be a monoid. We simply have to decide what happens when someone types a list of length more than 32.

One option is to say that the last character typed overwrites the 32nd entry,

$$[a_1, a_2, \dots, a_{31}, a_{32}, b] \sim_1 [a_1, a_2, \dots, a_{31}, b].$$

Another option is to say that any character typed after_32 entries is discarded,

$$[a_1, a_2, \dots, a_{31}, a_{32}, b] \sim_2 [a_1, a_2, \dots, a_{31}, a_{32}].$$

Both of these yield finitely presented monoids, generated by G. (In case it's useful, the number of necessary relations in both cases is 26^{33}.)

$\lozenge\lozenge$

Exercise 3.1.1.21. Let's consider the buffer concept again (see Application 3.1.1.20), but this time only having size 3 rather than size 32. Show using Definition 3.1.1.17 that with relations given by $\sim_1$ we indeed have $[a,b,c,d,e,f] = [a,b,f]$ and that with relations given by $\sim_2$ we indeed have $[a,b,c,d,e,f] = [a,b,c]$. $\lozenge$

Exercise 3.1.1.22. Let $K := \{BS, a, b, c, \dots, z\}$, a set having 27 elements. Suppose you want to think of $BS \in K$ as the "backspace key" and the elements $a, b, \dots z \in K$ as the letter keys on a keyboard. Then the free monoid $\text{List}(K)$ is not quite appropriate as a model because we want $[a,b,d,BS] = [a,b]$.

a.) Choose a set of relations for which the monoid presented by generators K and the chosen relations is appropriate to this application.

b.) Under your relations, how does $[BS]$ compare with $[\,]$? Is that suitable?

$\lozenge$

3.1.1.23 Cyclic monoids

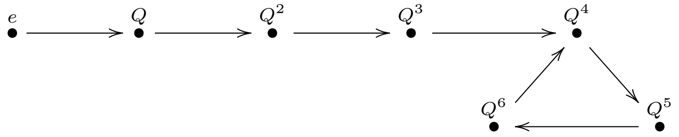
Definition 3.1.1.24. A monoid is called *cyclic* if it has a presentation involving only one generator.

Example 3.1.1.25. Let Q be a symbol; we look at some cyclic monoids generated by $\{Q\}$. With no relations the monoid would be the free monoid on one generator, and would have underlying set $\{[\,], [Q], [Q,Q], [Q,Q,Q], \dots\}$, with identity element $[\,]$ and multiplication given by concatenation (e.g. $[Q,Q,Q] +\!\!+ [Q,Q] = [Q,Q,Q,Q,Q]$). This is just $\mathbb{N}$, the additive monoid of natural numbers.

With the really strong relation $[Q] \sim [\,]$ we would get the trivial monoid, a monoid having only one element (see Example 3.1.1.10).

Another possibility is given in the first part of Example 3.1.2.3, where the relation $Q^{12} \sim [\,]$ is used, where Q^{12} is shorthand for $[Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q,Q]$.

Example 3.1.1.26. Consider the cyclic monoid with generator Q and relation $Q^7 = Q^4$. This monoid has seven elements, $\{e = Q^0, Q = Q^1, Q^2, Q^3, Q^4, Q^5, Q^6\}$, and we know that $Q^6 \star Q^5 = Q^7 * Q^4 = Q^4 * Q^4 = Q^7 * Q = Q^5$. One might depict this monoid as follows



To see the mathematical source of this intuitive depiction, see Example 5.2.1.17.

Exercise 3.1.1.27 (Classify the cyclic monoids). Classify all the cyclic monoids up to isomorphism. That is, come up with a naming system such that every cyclic monoid can be given a name in your system, such that no two non-isomorphic cyclic monoids have the same name, and such that no name exists in the system unless it refers to a cyclic monoid.

Hint: one might see a pattern in which the three monoids in Example 3.1.1.25 correspond respectively to ∞, 1, and 12, and then think "Cyclic monoids can be classified by (i.e. systematically named by elements of) the set $\mathbb{N} \sqcup \{\infty\}$." That idea is on the right track, but is not correct. $\Diamond$

3.1.2 Monoid actions

Definition 3.1.2.1 (Monoid action)**.** Let $(M, e, \star)$ be a monoid and let S be a set. An *action of* $(M, e, \star)$ *on* S, or simply an *action of* M *on* S or an M*-action on* S, is a function

$$\circlearrowleft\ : M \times S \to S$$

such that the following conditions hold for all $m, n \in M$ and all $s \in S$:

- $e \circlearrowleft s = s$
- $m \circlearrowleft (n \circlearrowleft s) = (m \star n) \circlearrowleft s$. [4]

Remark 3.1.2.2. To be pedantic (and because it's sometimes useful), we may rewrite $\circlearrowleft$ as $\alpha \colon M \times S \to S$ and restate the conditions from Definition 3.1.2.1 as

- $\alpha(e, s) = s$, and
- $\alpha(m, \alpha(n, s)) = \alpha(m \star n, s)$.

Example 3.1.2.3. Let $S = \{0, 1, 2, \ldots, 11\}$ and let $N = (\mathbb{N}, 0, +)$ be the additive monoid of natural numbers (see Example 3.1.1.3). We define a function $\circlearrowleft \colon \mathbb{N} \times S \to S$ by taking a pair (n, s) to the remainder that appears when $n + s$ is divided by 12. For example $4 \circlearrowleft 2 = 6$ and $8 \circlearrowleft 9 = 5$. This function has the structure of a monoid action because the two rules from Definition 3.1.2.1 hold.

[4] Definition 3.1.2.1 actually defines a *left action* of $(M, e, \star)$ on S. A *right action* is like a left action except the order of operations is somehow reversed. We will not really use right-actions in this text, but we briefly define it here for completeness. With notation as above, the only difference is in the second condition. We replace it by the condition that for all $m, n \in M$ and all $s \in S$ we have

$$m \circlearrowleft (n \circlearrowleft s) = (n \star m) \circlearrowleft s$$

Similarly, let T denote the set of points on a circle, elements of which are denoted by a real number in the interval $[0, 12)$, i.e.

$$T = \{x \in \mathbb{R} \mid 0 \leqslant x < 12\}$$

and let $R = (\mathbb{R}, 0, +)$ denote the additive monoid of real numbers. Then there is an action $R \times T \to T$, similar to the one above (see Exercise 3.1.2.4).

One can think of this as an action of the monoid of time on the clock.

Exercise 3.1.2.4.

a.) Realize the set $T := [0, 12) \subseteq \mathbb{R}$ as the coequalizer of a pair of arrows $\mathbb{R} \rightrightarrows \mathbb{R}$.

b.) For any $x \in \mathbb{R}$, realize the mapping $x \cdot - : T \to T$, implied by Example 3.1.2.3, using the universal property of coequalizers.

c.) Prove that it is an action.

◊

Exercise 3.1.2.5. Let B denote the set of buttons (or positions) of a video game controller (other than, say 'start' and 'select'), and consider the free monoid $\mathsf{List}(B)$ on B.

a.) What would it mean for $\mathsf{List}(B)$ to act on the set of states of some game? Imagine a video game G' that uses the controller, but for which $\mathsf{List}(B)$ would not be said to act on the states of G'. Now imagine a simple game G for which $\mathsf{List}(B)$ would be said to act.

b.) Can you think of a state s of G, and two distinct elements $\ell, \ell' \in \mathsf{List}(B)$ such that $\ell \circlearrowright s = \ell' \circlearrowright s$? In video game parlance, what would you call an element $b \in B$ such that, for every state $s \in G$, one has $b \circlearrowright s = s$?

c.) In video game parlance, what would you call a state $s \in S$ such that, for every sequence of buttons $\ell \in \mathsf{List}(B)$, one has $\ell \circlearrowright s = s$?

◊

Application 3.1.2.6. Let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function of which we want to find roots (points $x \in \mathbb{R}$ such that $f(x) = 0$). Let $x_0 \in \mathbb{R}$ be a starting point. For any $n \in \mathbb{N}$ we can apply Newton's method to x_n to get

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

This is a monoid (namely $\mathbb{N}$, the free monoid on one generator) acting on a set (namely $\mathbb{R}$).

However, Newton's method can get into trouble. For example at a critical point it causes division by 0, and sometimes it can oscillate or overshoot. In these cases we want to perturb a bit to the left or right. To have these actions available to us, we would add "perturb" elements to our monoid. Now we have more available actions at any point, but at the cost of using a more complicated monoid.

When publishing an experimental finding, there may be some deep methodological questions that are not considered suitably important to mention. For example, one may not publish the kind solution finding method (e.g. Newton's method or Runge-Kutta) that was used, nor the set of available actions, e.g. what kinds of perturbation were used by the researcher. However, these may actually influence the reproducibility of results.

By using a language such as that of monoid actions, we can align our data model with our unspoken assumptions about how functions are analyzed.

◊◊

Remark 3.1.2.7. A monoid is useful for understanding how an agent acts on the set of states of an object, but there is only one *kind* of action. At any point, all actions are available. In reality it is often the case that contexts can change and different actions are available at different times. For example on a computer, the commands available in one application have no meaning in another. This will get us to categories in the next chapter.

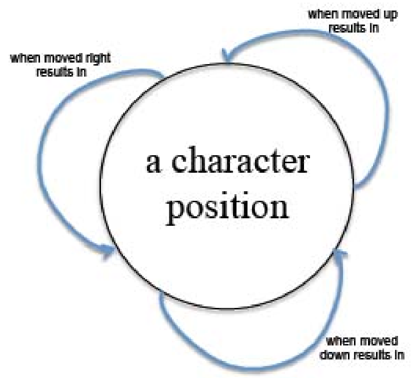
3.1.2.8 Monoids actions as ologs

If monoids are understood in terms of how they act on sets, then it is reasonable to think of them in terms of ologs. In fact, the ologs associated to monoids are precisely those ologs that have exactly one type (and possibly many arrows and commutative diagrams).

Example 3.1.2.9. In this example we show how to associate an olog to a monoid action. Consider the monoid M generated by the set $\{u, d, r\}$, standing for "up, down, right", and subject to the relations

$$[u, d] \sim [\,], \qquad [d, u] \sim [\,], \qquad [u, r] = [r, u], \qquad \text{and} \qquad [d, r] = [r, d].$$

We might imagine that M acts on the set of positions for a character in an old video game. In that case the olog corresponding to this action should look something like the following:



Given x, a character position, consider the following. We know that x is a character position, which when moved up results in a character position, which when moved down results in a character position that we'll call P(x). We also know that x is a character position that we'll call Q(x). Fact: whenever x is a character position we will have P(x)=Q(x). **Summary: [up, down] = [ ]**

Given x, a character position, consider the following. We know that x is a character position, which when moved down results in a character position, which when moved up results in a character position that we'll call P(x). We also know that x is a character position that we'll call Q(x). Fact: whenever x is a character position we will have P(x)=Q(x). **Summary: [down, up] = [ ]**

Given x, a character position, consider the following. We know that x is a character position, which when moved up results in a character position, which when moved right results in a character position that we'll call P(x). We also know that x is a character position, which when moved right results in a character position, which when moved up results in a character position that we'll call Q(x). Fact: whenever x is a character position we will have P(x)=Q(x). **Summary: [up, right] = [right, up]**

Given x, a character position, consider the following. We know that x is a character position, which when moved down results in a character position, which when moved right results in a character position that we'll call P(x). We also know that x is a character position, which when moved right results in a character position, which when moved down results in a character position that we'll call Q(x). Fact: whenever x is a character position we will have P(x)=Q(x). **Summary: [down, right] = [right, down]**

3.1.2.10 Finite state machines

According to Wikipedia, a *deterministic finite state machine* is a quintuple $(\Sigma, S, s_0, \delta, F)$, where

1. Σ is a finite non-empty set of symbols, called the *input alphabet*,

2. S is a finite, non-empty set, called *the state set*,

3. $\delta\colon \Sigma \times S \to S$ is a function, called the *state-transition function*, and

4. $s_0 \in S$ is an element, called *the initial state*,

5. $F \subseteq S$ is a subset, called the *set of final states.*

In this book we will not worry about the initial state and the set of final states, concerning ourselves more with the interaction via δ of the alphabet Σ on the set S of states.

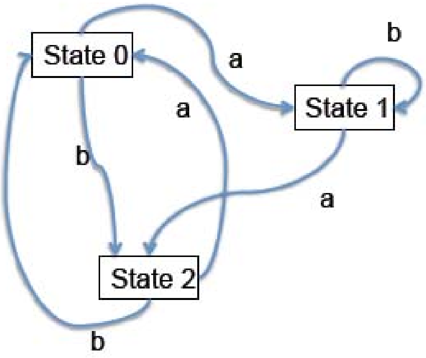


Figure 3.1: A finite state machine with alphabet $\Sigma = \{a, b\}$ and state set $S = \{\text{State 0, State 1, State 2}\}$. If pressed, we will make State 0 the initial state and $\{\text{State 2}\}$ the set of final states.

The following proposition expresses the notion of finite state automata in terms of free monoids and their actions on finite sets.

Proposition 3.1.2.11. *Let Σ, S be finite non-empty sets. Giving a function $\delta\colon \Sigma \times S \to S$ is equivalent to giving an action of the free monoid* $\text{List}(\Sigma)$ *on S.*

Proof. By Definition 3.1.2.1, we know that function $\epsilon\colon \text{List}(\Sigma) \times S \to S$ constitutes an action of the monoid $\text{List}(\Sigma)$ on the set S if and only if, for all $s \in S$ we have $\epsilon([\,], s) = s$, and for any two elements $m, m' \in \text{List}(\Sigma)$ we have $\epsilon(m, \epsilon(m', s)) = \epsilon(m \star m', s)$, where $m \star m'$ is the concatenation of lists. Let

$$A = \{\epsilon\colon \text{List}(\Sigma) \times S \to S \mid \epsilon \text{ constitutes an action}\}.$$

We need to prove that there is an isomorphism of sets

$$\phi\colon A \xrightarrow{\cong} \text{Hom}_{\textbf{Set}}(\Sigma \times S, S).$$

Given an element $\epsilon\colon \text{List}(\Sigma) \times S \to S$ in A, define $\phi(\epsilon)$ on an element $(\sigma, s) \in \Sigma \times S$ by $\phi(\epsilon)(\sigma, s) := \epsilon([\sigma], s)$, where $[\sigma]$ is the one-element list. We now define $\psi\colon \text{Hom}_{\textbf{Set}}(\Sigma \times S, S) \to A$.

Given an element $f \in \text{Hom}_{\textbf{Set}}(\Sigma \times S, S)$, define $\psi(f)\colon \text{List}(\Sigma) \times S \to S$ on a pair $(L, s) \in \text{List}(\Sigma) \times S$, where $L = [\epsilon_1, \ldots, \epsilon_n]$ as follows. By induction, if $n = 0$, put $\psi(f)(L, s) = s$; if $n \geqslant 1$, let $L' = [\epsilon_1, \ldots, \epsilon_{n-1}]$ and put $\psi(f)(L, s) = \psi(f)(L', f(\epsilon_n, s))$. One checks easily that $\psi(f)$ satisfies the two rules above, making it an action of $\text{List}(\Sigma)$ on S. It is also easy to check that ϕ and ψ are mutually inverse, completing the proof. □

We sum up the idea of this section as follows:

Slogan 3.1.2.12.

> “ *A finite state machine is an action of a free monoid on a finite set.* ”

Exercise 3.1.2.13. Consider the functions ϕ and ψ above.

a.) Show that for any $f\colon \Sigma \times S \to S$, the map $\psi(f)\colon \mathrm{List}(\Sigma) \times S \to S$ constitutes an action.

b.) Show that ϕ and ψ are mutually inverse functions (i.e. $\phi \circ \psi = \mathrm{id}_{\mathrm{Hom}(\Sigma \times S, S)}$ and $\psi \circ \phi = \mathrm{id}_A$.)

◊

3.1.3 Monoid action tables

Let M be a monoid generated by the set $G = \{g_1, \ldots, g_m\}$, and with some relations, and suppose that $\alpha\colon M \times S \to S$ is an action of M on a set $S = \{s_1, \ldots, s_n\}$. We can represent the action α using an *action table* whose columns are the elements of G and whose rows are the elements of S. In each cell $(\mathit{row}, \mathit{col})$, where $\mathit{row} \in S$ and $\mathit{col} \in G$, we put the element $\alpha(\mathit{col}, \mathit{row}) \in S$.

Example 3.1.3.1 (Action table). If Σ and S are the sets from Figure 3.1, the displayed action of $\mathrm{List}(\Sigma)$ on S would be given by the action table

Action from 3.1		
ID	**a**	**b**
State 0	State 1	State 2
State 1	State 2	State 1
State 2	State 0	State 0

(3.2)

Example 3.1.3.2 (Multiplication action table). Every monoid acts on itself by its multiplication formula, $M \times M \to M$. If G is a generating set for M, we can write the elements of G as the columns and the elements of M as rows, and call this a multiplication table. For example, let $(\mathbb{N}, 1, *)$ denote the multiplicative monoid of natural numbers. The multiplication table is as follows:

Multiplication of natural numbers							
$\mathbb{N}$	**0**	**1**	**2**	**3**	**4**	**5**	$\cdots$
0	0	0	0	0	0	0	$\cdots$
1	0	1	2	3	4	5	$\cdots$
2	0	2	4	6	8	10	$\cdots$
3	0	3	6	9	12	15	$\cdots$
4	0	4	8	12	16	20	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
21	0	21	42	63	84	105	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

(3.3)

Try to understand what is meant by this: “applying column 2 and then column 2 returns the same thing as applying column 4.”

In the above table, we were implicitly taking every element of $\mathbb{N}$ as a generator (since we had a column for every natural number). In fact, there is a smallest generating set for the monoid $(\mathbb{N}, 1, *)$, so that every element of the monoid is a product of some combination of these generators, namely the primes and 0.

Multiplication of natural numbers							
$\mathbb{N}$	$\mathbf{0}$	$\mathbf{2}$	$\mathbf{3}$	$\mathbf{5}$	$\mathbf{7}$	$\mathbf{11}$	$\cdots$
0	0	0	0	0	0	0	$\cdots$
1	0	2	3	5	7	11	$\cdots$
2	0	4	6	10	14	22	$\cdots$
3	0	6	9	15	21	33	$\cdots$
4	0	8	12	20	28	44	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
21	0	42	63	105	147	231	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Exercise 3.1.3.3. Let $\mathbb{N}$ be the additive monoid of natural numbers, let $S = \{0, 1, 2, \ldots, 11\}$, and let $\cdot\colon \mathbb{N} \times S \to S$ be the action given in Example 3.1.2.3. Using a nice small generating set for the monoid, write out the corresponding action table. ◊

3.1.4 Monoid homomorphisms

A monoid $(M, e, \star)$ involves a set, an identity element, and a multiplication formula. For two monoids to be comparable, their sets, their identity elements, and their multiplication formulas should be appropriately comparable. For example the additive monoids $\mathbb{N}$ and $\mathbb{Z}$ should be comparable because $\mathbb{N} \subseteq \mathbb{Z}$ is a subset, the identity elements in both cases are the same $e = 0$, and the multiplication formulas are both integer addition.

Definition 3.1.4.1. Let $\mathcal{M} := (M, e, \star)$ and $\mathcal{M}' := (M', e', \star')$ be monoids. A *monoid homomorphism f from $\mathcal{M}$ to $\mathcal{M}'$*, denoted $f\colon \mathcal{M} \to \mathcal{M}'$, is a function $f\colon M \to M'$ satisfying two conditions:

- $f(e) = e'$, and
- $f(m_1 \star m_2) = f(m_1) \star' f(m_2)$, for all $m_1, m_2 \in M$.

The set of monoid homomorphisms from $\mathcal{M}$ to $\mathcal{M}'$ is denoted $\mathrm{Hom}_{\mathbf{Mon}}(\mathcal{M}, \mathcal{M}')$.

Example 3.1.4.2 (From $\mathbb{N}$ to $\mathbb{Z}$). As stated above, the inclusion map $i\colon \mathbb{N} \to \mathbb{Z}$ induces a monoid homomorphism $(\mathbb{N}, 0, +) \to (\mathbb{Z}, 0, +)$ because $i(0) = 0$ and $i(n_1 + n_2) = i(n_1) + i(n_2)$.

Let $i_5\colon \mathbb{N} \to \mathbb{Z}$ denote the function $i_5(n) = 5 * n$, so $i_5(4) = 20$. This is also a monoid homomorphism because $i_5(0) = 5 * 0 = 0$ and $i_5(n_1 + n_2) = 5 * (n_1 + n_2) = 5 * n_1 + 5 * n_2 = i_5(n_1) + i_5(n_2)$.

Application 3.1.4.3. Let $R = \{a, c, g, u\}$ and let $T = R^3$, the set of triplets in R. Let $\mathcal{R} = \mathrm{List}(R)$ be the free monoid on R and let $\mathcal{T} = \mathrm{List}(T)$ denote the free monoid on

T. There is a monoid homomorphism $F\colon \mathcal{T} \to \mathcal{R}$ given by sending $t = (r_1, r_2, r_3)$ to the list $[r_1, r_2, r_3]$. [5]

If A be the set of amino acids and $\mathcal{A} = \mathrm{List}(A)$ the free monoid on A, the process of translation gives a monoid homomorphism $G\colon \mathcal{T} \to \mathcal{A}$, turning a list of RNA triplets into a polypeptide. But how do we go from a list of RNA nucleotides to a polypeptide? The answer is that there is no good way to do this mathematically. So what is going wrong?

The answer is that there should not be a monoid homomorphism $\mathcal{R} \to \mathcal{A}$ because not all sequences of nucleotides produce a polypeptide; for example if the sequence has only two elements, it does not code for a polypeptide. There are several possible remedies to this problem. One is to take the image of F, which is a submonoid $\mathcal{R}' \subseteq \mathcal{R}$. It is not hard to see that there is a monoid homomorphism $F'\colon \mathcal{R}' \to \mathcal{T}$, and we can compose it with G to get our desired monoid homomorphism $G \circ F'\colon \mathcal{R}' \to \mathcal{A}$. [6]

◊◊

Example 3.1.4.4. Given any monoids $\mathcal{M}$ there is a unique monoid homomorphism from $\mathcal{M}$ to the trivial monoid $\underline{1}$ (see Example 3.1.1.10). There is also a unique homomorphism $\underline{1} \to \mathcal{M}$. These facts together have an upshot: between any two monoids $\mathcal{M}$ and $\mathcal{M}'$ we can always construct a homomorphism

$$\mathcal{M} \xrightarrow{!} \underline{1} \xrightarrow{!} \mathcal{M}'$$

which we call the *trivial homomorphism* $\mathcal{M} \to \mathcal{M}'$. A morphism $\mathcal{M} \to \mathcal{M}'$ that is not trivial is called a *nontrivial homomorphism.*

Proposition 3.1.4.5. *Let $\mathcal{M} = (\mathbb{Z}, 0, +)$ and $\mathcal{M}' = (\mathbb{N}, 0, +)$. The only monoid homomorphism $f\colon \mathcal{M} \to \mathcal{M}'$ sends every element $m \in \mathbb{Z}$ to $0 \in \mathbb{N}$.*

Proof. Let $f\colon \mathcal{M} \to \mathcal{M}'$ be a monoid homomorphism, and let $n = f(1)$ and $n' = f(-1)$ in $\mathbb{N}$. Then we know that since $0 = 1+(-1)$ in $\mathbb{Z}$ we must have $0 = f(0) = f(1+(-1)) = f(1) + f(-1) = n + n' \in \mathbb{N}$. But if $n \geqslant 1$ then this is impossible, so $n = 0$. Similarly $n' = 0$. Any element $m \in \mathbb{Z}$ can be written $m = 1+1+\cdots+1$ or as $m = -1+-1+\cdots+-1$, and it is easy to see that $f(1) + f(1) + \cdots + f(1) = 0 = f(-1) + f(-1) + \cdots + f(-1)$. Therefore, $f(m) = 0$ for all $m \in \mathbb{Z}$.

□

Exercise 3.1.4.6. For any $m \in \mathbb{N}$ let $i_m\colon \mathbb{N} \to \mathbb{Z}$ be the function $i_m(n) = m * n$. All such functions are monoid homomorphisms $(\mathbb{N}, 0, +) \to (\mathbb{Z}, 0, +)$. Do any monoid homomorphisms $(\mathbb{N}, 0, +) \to (\mathbb{Z}, 0, +)$ not come in this way? For example, what about using $n \mapsto 5 * n - 1$ or $n \mapsto n^2$, or some other function? ◊

Exercise 3.1.4.7. Let $\mathcal{M} := (\mathbb{N}, 0, +)$ be the additive monoid of natural numbers, let $\mathcal{N} = (\mathbb{R}_{\geqslant 0}, 0, +)$ be the additive monoid of nonnegative real numbers, and let $\mathcal{P} := (\mathbb{R}_{>0}, 1, *)$ be the multiplicitive monoid of positive real numbers. Can you think of any nontrivial monoid homomorphisms of the following sorts:

$$\mathcal{M} \to \mathcal{N}, \qquad \mathcal{M} \to \mathcal{P}, \qquad \mathcal{N} \to \mathcal{P}, \qquad \mathcal{N} \to \mathcal{M}, \qquad \mathcal{P} \to \mathcal{N}?$$

◊

[5] More precisely, the monoid homomorphism F sends a list $[t_1, t_2, \ldots, t_n]$ to the list $[r_{1,1}, r_{1,2}, r_{1,3}, r_{2,1}, r_{2,2}, r_{2,3}, \ldots, r_{n,1}, r_{n,2}, r_{n,3}]$, where for each $0 \leqslant i \leqslant n$ we have $t_i = (r_{i,1}, r_{i,2}, r_{i,3})$.

[6] Adding stop-codons to the mix we can handle more of $\mathcal{R}$, e.g. sequences that don't have a multiple-of-three many nucleotides.

3.1.4.8 Homomorphisms from free monoids

Recall that $(\mathbb{N}, 0, +)$ is the free monoid on one generator. It turns out that for any other monoid $\mathcal{M} = (M, e, \star)$, the set of monoid homomorphisms $\mathbb{N} \to \mathcal{M}$ is in bijection with the set M. This is a special case (in which G is a set with one element) of the following proposition.

Proposition 3.1.4.9. *Let G be a set, let $F(G) := (\mathrm{List}(G), [\,], +\!\!+)$ be the free monoid on G, and let $\mathcal{M} := (M, e, \star)$ be any monoid. There is a natural bijection*

$$\mathrm{Hom}_{\mathbf{Mon}}(F(G), \mathcal{M}) \xrightarrow{\cong} \mathrm{Hom}_{\mathbf{Set}}(G, M).$$

Proof. We provide a function $\phi\colon \mathrm{Hom}_{\mathbf{Mon}}(F(G), \mathcal{M}) \to \mathrm{Hom}_{\mathbf{Set}}(G, M)$ and a function $\psi\colon \mathrm{Hom}_{\mathbf{Set}}(G, M) \to \mathrm{Hom}_{\mathbf{Mon}}(F(G), \mathcal{M})$ and show that they are mutually inverse. Let us first construct ϕ. Given a monoid homomorphism $f\colon F(G) \to \mathcal{M}$, we need to provide $\phi(f)\colon G \to M$. Given any $g \in G$ we define $\phi(f)(g) := f([g])$.

Now let us construct ψ. Given $p\colon G \to M$, we need to provide $\psi(p)\colon \mathrm{List}(G) \to \mathcal{M}$ such that $\psi(p)$ is a monoid homomorphism. For a list $L = [g_1, \ldots, g_n] \in \mathrm{List}(G)$, define $\psi(p)(L) := p(g_1) \star \cdots \star p(g_n) \in M$. In particular, $\psi(p)([\,]) = e$. It is not hard to see that this is a monoid homomorphism. It is also easy to see that $\phi \circ \psi(p) = p$ for all $p \in \mathrm{Hom}_{\mathbf{Set}}(G, M)$. We show that $\psi \circ \phi(f) = f$ for all $f \in \mathrm{Hom}_{\mathbf{Mon}}(F(G), \mathcal{M})$. Choose $L = [g_1, \ldots, g_n] \in \mathrm{List}(G)$. Then

$$\psi(\phi f)(L) = (\phi f)(g_1) \star \cdots \star (\phi f)(g_n) = f[g_1] \star \cdots \star f[g_n] = f([g_1, \ldots, g_n]) = f(L).$$

□

Exercise 3.1.4.10. Let $G = \{a, b\}$, let $\mathcal{M} := (M, e, \star)$ be any monoid, and let $f\colon G \to M$ be given by $f(a) = m$ and $f(b) = n$, where $m, n \in M$. If $\psi\colon \mathrm{Hom}_{\mathbf{Set}}(G, M) \to \mathrm{Hom}_{\mathbf{Mon}}(F(G), \mathcal{M})$ is the function from the proof of Proposition 3.1.4.9 and $L = [a, a, b, a, b]$, what is $\psi(f)(L)$? ◊

3.1.4.11 Restriction of scalars

A monoid homomorphism $f\colon M \to M'$ (see Definition 3.1.4.1) ensures that the elements of M have a reasonable interpretation in M'; they act the same way over in M' as they did back home in M. If we have such a homomorphism f and we have an action $\alpha\colon M' \times S \to S$ of M' on a set S, then we have a method for allowing M to act on S as well. Namely, we take an element of M, send it over to M', and act on S. In terms of functions, we compose α with the function $f \times \mathrm{id}_S\colon M \times S \to M' \times S$, to get a function we'll denote

$$\Delta_f(\alpha)\colon M \times S \to S.$$

After Proposition 3.1.4.12 we will know that $\Delta_f(\alpha)$ is indeed a monoid action, and we say that it is given by *restriction of scalars along f*.

Proposition 3.1.4.12. *Let $\mathcal{M} := (M, e, \star)$ and $\mathcal{M}' := (M', e', \star')$ be monoids, $f\colon \mathcal{M} \to \mathcal{M}'$ a monoid homomorphism, S a set, and suppose that $\alpha\colon M' \times S \to S$ is an action of $\mathcal{M}'$ on S. Then $\Delta_f(\alpha)\colon M \times S \to S$, defined as above, is a monoid action as well.*

Proof. Refer to Remark 3.1.2.2; we assume α is a monoid action and want to show that $\Delta_f(\alpha)$ is too. We have $\Delta_f(\alpha)(e, s) = \alpha(f(e), s) = \alpha(e', s) = s$. We also have

$$\begin{aligned}\Delta_f(\alpha)(m, \Delta_f(\alpha)(n, s)) = \alpha(f(m), \alpha(f(n), s)) &= \alpha(f(m) \star' f(n), s)\\ &= \alpha(f(m \star n), s)\\ &= \Delta_f(\alpha)(m \star n, s).\end{aligned}$$

□

Example 3.1.4.13. Let $\mathbb{N}$ and $\mathbb{Z}$ denote the additive monoids of natural numbers and integers, respectively, and let $i\colon \mathbb{N} \to \mathbb{Z}$ be the inclusion, which we saw in Example 3.1.4.2 is a monoid homomorphism. There is an action $\alpha\colon \mathbb{Z} \times \mathbb{R} \to \mathbb{R}$ of the monoid $\mathbb{Z}$ on the set $\mathbb{R}$ of real numbers, given by $\alpha(n, x) = n + x$. Clearly, this action works just as well if we restrict our scalars to $\mathbb{N} \subseteq \mathbb{Z}$, allowing ourselves only to add natural numbers to reals. The action $\Delta_i\alpha\colon \mathbb{N} \times \mathbb{R} \to \mathbb{R}$ is given on $(n, x) \in \mathbb{N} \times \mathbb{R}$ by $\Delta_i\alpha(n, x) = \alpha(i(n), x) = \alpha(n, x) = n + x$, just as expected.

Example 3.1.4.14. Suppose that V is a complex vector space. In particular, this means that the monoid $\mathbb{C}$ of complex numbers (under multiplication) acts on the elements of V. If $i\colon \mathbb{R} \to \mathbb{C}$ is the inclusion of the real line inside $\mathbb{C}$, then i is a monoid homomorphism. Restriction of scalars in the above sense turns V into a real vector space, so the name "restriction of scalars" is apt.

Exercise 3.1.4.15. Let $\mathbb{N}$ be the free monoid on one generator, let $\Sigma = \{a, b\}$, and let $S = \{\text{State 0, State 1, State 2}\}$. Consider the map of monoids $f\colon \mathbb{N} \to \text{List}(\Sigma)$ given by sending $1 \mapsto [a, b, b]$. The monoid action $\alpha\colon \text{List}(\Sigma) \times S \to S$ given in Example 3.1.3.1 can be transformed by restriction of scalars along f to an action $\Delta_f(\alpha)$ of $\mathbb{N}$ on S. Write down its action table. ◊

3.2 Groups

Groups are monoids in which every element has an inverse. If we think of these structures in terms of how they act on sets, the difference between groups and monoids is that the action of every group element can be undone. One way of thinking about groups is in terms of symmetries. For example, the rotations and reflections of a square form a group.

Another way to think of the difference between monoids and groups is in terms of time. Monoids are likely useful in thinking about diffusion, in which time plays a role and things cannot be undone. Groups are more likely useful in thinking about mechanics, where actions are time-reversible.

3.2.1 Definition and examples

Definition 3.2.1.1. Let $(M, e, \star)$ be a monoid. An element $m \in M$ is said to *have an inverse* if there exists an $m' \in M$ such that $mm' = e$ and $m'm = e$. A *group* is a monoid $(M, e, \star)$ in which every element $m \in M$ has an inverse.

Proposition 3.2.1.2. *Suppose that $\mathcal{M} := (M, e, \star)$ is a monoid and let $m \in M$ be an element. Then m has at most one inverse.* [7]

[7] If $\mathcal{M}$ is a group then every element m has exactly one inverse.

Proof. Suppose that both m' and m'' are inverses of m; we want to show that $m' = m''$. This follows by the associative law for monoids:

$$m' = m'(mm'') = (m'm)m'' = m''.$$

□

Example 3.2.1.3. The additive monoid $(\mathbb{N}, 0, +)$ is not a group because none of its elements are invertible, except for 0. However, the monoid of integers $(\mathbb{Z}, 0, +)$ is a group. The monoid of clock positions from Example 3.1.1.25 is also a group. For example the inverse of Q^5 is Q^7 because $Q^5 \star Q^7 = e = Q^7 \star Q^5$.

Example 3.2.1.4. Consider a square centered at the origin in $\mathbb{R}^2$. It has rotational and mirror symmetries. There are eight of these, which we denote

$$\{e, \rho, \rho^2, \rho^3, \phi, \phi\rho, \phi\rho^2, \phi\rho^3\},$$

where ρ stands for 90° counterclockwise rotation and ϕ stands for horizontal-flip (across the vertical axis). So relations include $\rho^4 = e$, $\phi^2 = e$, and $\rho^3\phi = \phi\rho$.

Example 3.2.1.5. The set of 3×3 matrices can be given the structure of a monoid, where the identity element is the 3×3 identity matrix, the multiplication is matrix multiplication. The subset of invertible matrices forms a group, called *the general linear group of dimension 3* and denoted GL_3. Inside of GL_3 is the so-called *orthogonal group*, denoted O_3, of matrices M such that $M^{-1} = M^\top$. These matrices correspond to symmetries of the sphere centered at the origin.

Another interesting group is the Euclidean group $E(3)$ which consists of all *isometries* of $\mathbb{R}^3$, i.e. all functions $\mathbb{R}^3 \to \mathbb{R}^3$ that preserve distances.

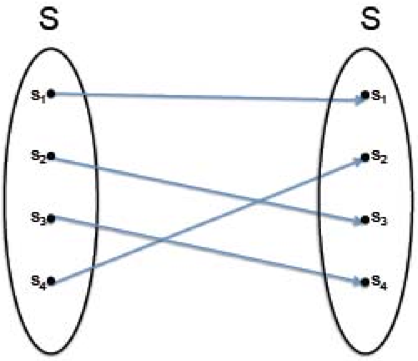
Application 3.2.1.6. In crystallography one is often concerned with the symmetries that arise in the arrangement A of atoms in a molecule. To think about symmetries in terms of groups, we first define an *atom-arrangement* to be a finite subset $i\colon A \subseteq \mathbb{R}^3$. A symmetry in this case is an isometry of $\mathbb{R}^3$ (see Example 3.2.1.5), say $f\colon \mathbb{R}^3 \to \mathbb{R}^3$ such that there exists a dotted arrow making the diagram below commute:

$$\begin{tikzcd}
A \arrow[r, dashed] \arrow[d, "i"'] & A \arrow[d, "i"] \\
\mathbb{R}^3 \arrow[r, "f"'] & \mathbb{R}^3
\end{tikzcd}$$

That is, it's an isometry of $\mathbb{R}^3$ such that each atom of A is sent to a position currently occupied by an atom of A. It is not hard to show that the set of such isometries forms a group, called the *space group* of the crystal.

◊◊

Exercise 3.2.1.7. Let S be a finite set. A *permutation of S* is an isomorphism $f\colon S \xrightarrow{\cong} S$.



a.) Come up with an identity, and a multiplication formula, such that the set of permutations of S forms a monoid.

b.) Is it a group?

◊

Exercise 3.2.1.8. In Exercise 3.1.1.27 you classified the cyclic monoids. Which of them are groups? ◊

Definition 3.2.1.9 (Group action)**.** Let $(G, e, \star)$ be a group and S a set. An *action* of G on S is a function $\circlearrowleft: G \times S \to S$ such that for all $s \in S$ and $g, g' \in G$, we have

- $e \circlearrowleft s = s$ and
- $g \circlearrowleft (g' \circlearrowleft s) = (g \star g') \circlearrowleft s$.

In other words, considering G as a monoid, it is an action in the sense of Definition 3.1.2.1.

Example 3.2.1.10. When a group acts on a set, it has the character of symmetry. For example, consider the group whose elements are angles θ. This group may be denoted $U(1)$ and is often formalized as the unit circle in $\mathbb{C}$ of complex numbers $z = a + bi$ such that $|z| = a^2 + b^2 = 1$. The set of such points is given the structure of a group $(U(1), e, \star)$ by defining the identity element to be $e := 1 + 0i$ and the group law to be complex multiplication. But for those unfamiliar with complex numbers, this is simply angle addition where we understand that $360° = 0°$. If $\theta_1 = 190°$ and $\theta_2 = 278°$, then $\theta_1 \star \theta_2 = 468° = 108°$. In the language of complex numbers, $z = e^{i\theta}$.

The group $U(1)$ acts on any set that we can picture as having rotational symmetry about a fixed axis, such as the earth around the north-south axis. We will define $S = \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 = 1\}$, the unit sphere, and understand the rotational action of $U(1)$ on S.

We first show that $U(1)$ acts on $\mathbb{R}^3$ by $\theta \circlearrowleft (x, y, z) = (x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta, z)$, or with matrix notation as

$$\theta \circlearrowleft (x, y, z) := (x, y, z) \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Trigonometric identities ensure that this is indeed an action.

In terms of action tables, we would need infinitely many columns to express this action. Here is a sample

Action of $U(1)$ on $\mathbb{R}^3$			
$\mathbb{R}^3$	$\theta = 45°$	$\theta = 90°$	$\theta = 100°$
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(1,0,0)	(.71,.71,0)	(0,1,0)	(-.17,.98,0)
(0,1,-4.2)	(-.71,.71,-4.2)	(-1,0,-4.2)	(-.98,-.17,-4.2)
(3,4,2)	(4.95,.71,2)	(-4,3,2)	(3.42,-3.65,2)
⋮	⋮	⋮	⋮

Finally, we are looking to see that the action preserves length so that if $(x, y, z) \in S$ then $\theta \circlearrowleft (x, y, z) \in S$; this way we will have confirmed that $U(1)$ indeed acts on S. The calculation begins by assuming $x^2 + y^2 + z^2 = 1$ and checks

$$(x \cos\theta + y \sin\theta)^2 + (-x \sin\theta + y \cos\theta)^2 + z^2 = x^2 + y^2 + z^2 = 1.$$

Exercise 3.2.1.11. Let X be a set and consider the group of permutations of X (see Exercise 3.2.1.7), which we will denote Σ_X. Find a canonical action of Σ_X on X. ◊

Definition 3.2.1.12. Let G be a group acting on a set X. For any point $x \in X$, the *orbit of* x, denoted Gx, is the set

$$Gx := \{x' \in X \mid \exists g \in G \text{ such that } gx = x'\}.$$

Application 3.2.1.13. Let S be the surface of the earth, understood as a sphere, and let $G = U(1)$ be the group of angles acting on S as in Example 3.2.1.10. The orbit of any point $p = (x, y, z) \in S$ is the set of points on the same latitude line as p.

One may also consider a small band around the earth, i.e. the set $A = \{(x, y, z) \mid 1.0 \leqslant x^2 + y^2 + z^2 \leqslant 1.05\}$. The action of $U(1) \circlearrowleft S$ extends to an action $U(1) \circlearrowleft A$. The orbits are latitude-lines-at-altitude. A simplifying assumption in climatology may be given by assuming that $U(1)$ acts on all currents in the atmosphere in an appropriate sense. That way, instead of considering movement within the whole space A, we only allow movement that behaves the same way throughout each orbit of the group action. ◊◊

Exercise 3.2.1.14.

a.) Consider the $U(1)$ action on $\mathbb{R}^3$ given in Example 3.2.1.10. Describe the set of orbits of this action.

b.) What are the orbits of the action of the permutation group $\Sigma_{\{1,2,3\}}$ on the set $\{1, 2, 3\}$? (See Exercise 3.2.1.11.)

◊

Exercise 3.2.1.15. Let G be a group and X a set on which G acts by $\circlearrowleft\colon G \times X \to X$. Is "being in the same orbit" an equivalence relation on X? ◊

Definition 3.2.1.16. Let G and G' be groups. A *group homomorphism* $f\colon G \to G'$ is defined to be a monoid homomorphism $G \to G'$, where G and G' are being regarded as monoids in accordance with Definition 3.2.1.1.

3.3 Graphs

In this course, unless otherwise specified, whenever we speak of graphs we are not talking about curves in the plane, such as parabolas, or pictures of functions generally. We are speaking of systems of vertices and arrows.

We will take our graphs to be *directed*, meaning that every arrow points *from* a vertex *to* a vertex; rather than merely connecting vertices, arrows will have direction. If a and b are vertices, there can be many arrows from a to b, or none at all. There can be arrows from a to itself. Here is the formal definition in terms of sets and functions.

3.3.1 Definition and examples

Definition 3.3.1.1. A *graph* G consists of a sequence $G := (V, A, src, tgt)$ where

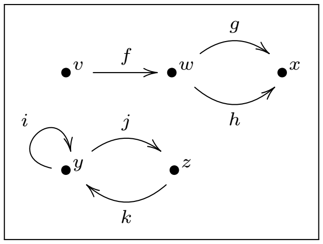
- V is a set, called *the set of vertices of* G (singular:*vertex*),
- A is a set, called *the set of arrows of* G,
- $src\colon A \to V$ is a function, called *the source function for* G, and
- $tgt\colon A \to V$ is a function, called *the target function for* G.

Given an arrow $a \in A$ we refer to $src(a)$ as the *source vertex* of a and to $tgt(a)$ as the *target vertex* of a.

To draw a graph, first draw a dot for every element of V. Then for every element $a \in A$, draw an arrow connecting dot $src(a)$ to dot $tgt(a)$.

Example 3.3.1.2 (Graph). Here is a picture of a graph $G = (V, A, src, tgt)$:

$G :=$



(3.4)

We have $V = \{v, w, x, y, z\}$ and $A = \{f, g, h, i, j, k\}$. The source and target functions $src, tgt\colon A \to V$ can be captured in the table to the left below:

A	**src**	**tgt**
f	v	w
g	w	x
h	w	x
i	y	y
j	y	z
k	z	y

V
v
w
x
y
z

In fact, all of the data of the graph G is captured in the two tables above—together they tell us the sets A and V and the functions src and tgt.

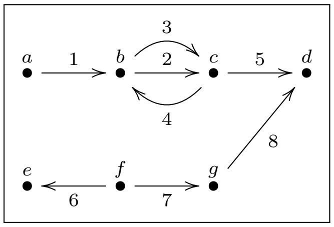
Example 3.3.1.3. Every olog has an underlying graph. The additional information in an olog has to do with which pairs of paths are declared equivalent, as well as text that has certain English-readability rules.

Exercise 3.3.1.4. a.) Draw the graph corresponding to the following tables:

A	**src**	**tgt**
f	v	w
g	v	w
h	v	w
i	x	w
j	z	w
k	z	z

V
u
v
w
x
y
z

b.) Write down two tables, as above, corresponding to the following graph:



◊

Exercise 3.3.1.5. Let $A = \{1, 2, 3, 4, 5\}$ and $B = \{a, b, c\}$. Draw them and choose an arbitrary function $f\colon A \to B$ and draw it. Let $A \sqcup B$ be the coproduct of A and B (Definition 2.4.2.1) and let $A \xrightarrow{i_1} A \sqcup B \xleftarrow{i_2} B$ be the two inclusions. Consider the two functions $src, tgt\colon A \to A \sqcup B$, where $src = i_1$ and tgt is the composition $A \xrightarrow{f} B \xrightarrow{i_2} A \sqcup B$. Draw the associated graph $(A \sqcup B, A, src, tgt)$. ◊

Exercise 3.3.1.6.

a.) Let V be a set. Suppose we just draw the elements of V as vertices and have no arrows between them. Is this a graph?

b.) Given V, is there any other "canonical" or somehow automatic non-random procedure for generating a graph with those vertices?

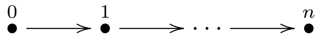
◊

Example 3.3.1.7. Recall from Construction 2.5.2.5 the notion of bipartite graph, which we defined to be a span (i.e. pair of functions, see Definition 2.5.2.1) $A \xleftarrow{f} R \xrightarrow{g} B$. Now that we have a formal definition of graph, we might hope that bipartite graphs fit in, and they do. Let $V = A \sqcup B$ and let $i\colon A \to V$ and $j\colon B \to V$ be the inclusions. Let $src = i \circ f\colon R \to V$ and let $tgt = j \circ g\colon R \to V$ be the composites.

$$\begin{tikzcd}
 & A \arrow[dr, "i"] & \\
R \arrow[ur, "f"] \arrow[rr, bend left, "src"] \arrow[rr, bend right, "tgt"'] \arrow[dr, "g"'] & & V \\
 & B \arrow[ur, "j"'] &
\end{tikzcd}$$

Then (V, R, src, tgt) is a graph that would be drawn exactly as we specified the drawing of spans in Construction 2.5.2.5.

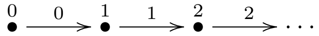
Example 3.3.1.8. Let $n \in \mathbb{N}$ be a natural number. The *chain graph of length n*, denoted $[n]$ is the graph depicted here:



In general $[n]$ has n arrows and $n+1$ vertices. In particular, when $n = 0$ we have that $[0]$ is the graph consisting of a single vertex and no arrows.

Example 3.3.1.9. Let $G = (V, A, src, tgt)$ be a graph; we want to spread it out over discrete time, so that each arrow does not occur within a given time-slice but instead over a quantum unit of time.

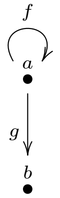
Let $N = (\mathbb{N}, \mathbb{N}, n \mapsto n, n \mapsto n+1)$ be the graph depicted



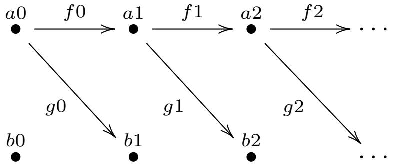
When we get to limits in a category, we will understand that products can be taken in the category of graphs (see Example 4.5.1.5), and $N \times G$ will make sense. For now, we construct it by hand.

Let $T(G) = (V \times \mathbb{N}, A \times \mathbb{N}, src', tgt')$ be a new graph, where for $a \in A$ and $n \in \mathbb{N}$ we have $src'(a, n) := (src(a), n)$ and $tgt'(a, n) = (tgt(a), n+1)$. This may be a bit much to swallow, so try to simply understand what is being done in the following example.

Let G be the graph drawn below

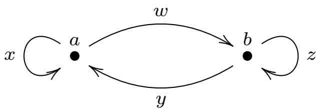


Then $T(G)$ will be the graph



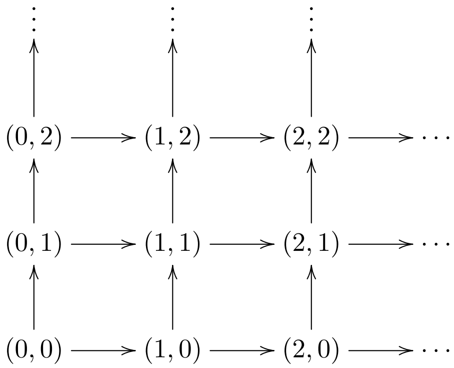
As you can see, f-arrows still take a's to a's and g-arrows still take a's to b's, but they always march forward in time.

Exercise 3.3.1.10. Let G be the graph depicted below:



Draw (using ellipses "$\cdots$" if necessary) the graph $T(G)$ defined in Example 3.3.1.9. ◊

Exercise 3.3.1.11. Consider the infinite graph $G = (V, A, src, tgt)$ depicted below,



a.) Write down the sets A and V.

b.) What are the source and target function $A \to V$?

◊

Exercise 3.3.1.12. A graph is a pair of functions $A \rightrightarrows V$. This sets up the notion of equalizer and coequalizer (see Definitions 2.5.3.1 and 2.6.3.1).

a.) What feature of a graph is captured by the equalizer of its source and target functions?

b.) What feature of a graph is captured by the coequalizer of its source and target functions?

◊

3.3.2 Paths in a graph

We all know what a path in a graph is, especially if we understand that a path must always follow the direction of arrows. The following definition makes this idea precise. In particular, one can have paths of any finite length $n \in \mathbb{N}$, even length 0 or 1. Also, we want to be able to talk about the source vertex and target vertex of a path, as well as concatenation of paths.

Definition 3.3.2.1. Let $G = (V, A, src, tgt)$ be a graph. A *path of length n* in G, denoted $p \in \text{Path}_G^{(n)}$ is a head-to-tail sequence

$$p = (v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \xrightarrow{a_3} \ldots \xrightarrow{a_n} v_n) \tag{3.5}$$

of arrows in G, which we denote by $v_0 a_1 a_2 \ldots a_n$. In particular we have canonical isomorphisms $\text{Path}_G^{(1)} \cong A$ and $\text{Path}_G^{(0)} \cong V$; we refer to the path of length 0 on vertex v as the *trivial path on v* and denote it simply by v. We denote by Path_G the set of paths in G,

$$\text{Path}_G := \bigcup_{n \in \mathbb{N}} \text{Path}_G^{(n)}.$$

Every path $p \in \text{Path}_G$ has a source vertex and a target vertex, and we may denote these by $\overline{src}, \overline{tgt} \colon \text{Path}_G \to V$. If p is a path with $\overline{src}(p) = v$ and $\overline{tgt}(p) = w$, we may denote it by $p\colon v \to w$. Given two vertices $v, w \in V$, we write $\mathrm{Path}_G(v,w)$ to denote the set of all paths $p\colon v \to w$.

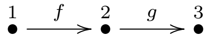
There is a concatenation operation on paths. Given a path $p\colon v \to w$ and $q\colon w \to x$, we define the concatenation, denoted $pq\colon v \to x$ in the obvious way. If $p = va_1, a_2 \ldots a_m$ and $q = wb_1b_2 \ldots b_n$ then $pq = va_1 \ldots a_m b_1 \ldots b_n$. In particular, if p (resp. r) is the trivial path on vertex v (resp. vertex w) then for any path $q\colon v \to w$, we have $pq = q$ (resp. $qr = q$).

Example 3.3.2.2. In Diagram (3.4), page 86, there are no paths from v to y, one path (f) from v to w, two paths (fg and fh) from v to x, and infinitely many paths

$$\{yi^{p_1}(jk)^{q_1} \cdots i^{p_n}(jk)^{q_n} \mid n, p_1, q_1, \ldots, p_n, q_n \in \mathbb{N}\}$$

from y to y. There are other paths as well, including the five trivial paths.

Exercise 3.3.2.3. How many paths are there in the following graph?



◊

Exercise 3.3.2.4. Let G be a graph and consider the set Path_G of paths in G. Suppose someone claimed that there is a monoid structure on the set Path_G, where the multiplication formula is given by concatenation of paths. Are they correct? Why or why not? Hint: what should be the identity element? ◊

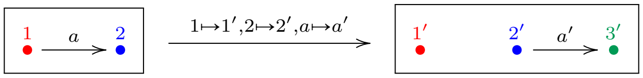
3.3.3 Graph homomorphisms

A graph (V, A, src, tgt) involves two sets and two functions. For two graphs to be comparable, their two sets and their two functions should be appropriately comparable.

Definition 3.3.3.1. Let $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ be graphs. A *graph homomorphism f from G to G'*, denoted $f\colon G \to G'$, consists of two functions $f_0\colon V \to V'$ and $f_1\colon A \to A'$ such that the two diagrams below commute:

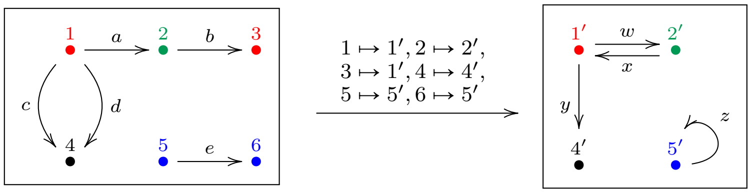
$$\begin{tikzcd} A \arrow[r, "f_1"] \arrow[d, "src"'] & A' \arrow[d, "src'"] \\ V \arrow[r, "f_0"'] & V' \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "f_1"] \arrow[d, "tgt"'] & A' \arrow[d, "tgt'"] \\ V \arrow[r, "f_0"'] & V' \end{tikzcd} \tag{3.6}$$

Remark 3.3.3.2. The above conditions (3.6) may look abstruse at first, but they encode a very important idea, roughly stated "arrows are bound to their vertices". Under a map of graphs $G \to G'$, one cannot flippantly send an arrow of G any old arrow of G': it must still connect the vertices it connected before. Below is an example of a mapping that does not respect this condition: a connects 1 and 2 before, but not after:



The commutativity of the diagrams in (3.6) is exactly what is needed to ensure that arrows are handled in the expected way by a proposed graph homomorphism.

Example 3.3.3.3 (Graph homomorphism). Let $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ be the graphs drawn to the left and right (respectively) below:



(3.7)

The colors indicate our choice of function $f_0\colon V \to V'$. Given that choice, condition (3.6) imposes in this case that there is a unique choice of graph homomorphism $f\colon G \to G'$.

Exercise 3.3.3.4.

a.) Where are a, b, c, d, e sent under $f_1\colon A \to A'$ in Diagram (3.7)?

b.) Choose a couple elements of A and check that they behave as specified by Diagram (3.6).

◊

Exercise 3.3.3.5. Let G be a graph, let $n \in \mathbb{N}$ be a natural number, and let $[n]$ be the chain graph of length n, as in Example 3.3.1.8. Is a path of length n in G the same thing as a graph homomorphism $[n] \to G$, or are there subtle differences? More precisely, is there always an isomorphism between the set of graph homomorphisms $[n] \to G$ and the set $\mathrm{Path}_G^{(n)}$ of length-n paths in G? ◊

Exercise 3.3.3.6. Given a morphism of graphs $f\colon G \to G'$, there an induced function $\mathrm{Path}(f)\colon \mathrm{Path}(G) \to \mathrm{Path}(G')$.

a.) Is it the case that for every $n \in \mathbb{N}$, the function $\mathrm{Path}(f)$ carries $\mathrm{Path}^{(n)}(G)$ to $\mathrm{Path}^{(n)}(G')$, or can path lengths change in this process?

b.) Suppose that f_0 and f_1 are injective (meaning no two distinct vertices in G are sent to the same vertex (respectively for arrows) under f). Does this imply that $\mathrm{Path}(f)$ is also injective (meaning no two distinct paths are sent to the same path under f)?

c.) Suppose that f_0 and f_1 are surjective (meaning every vertex in G' and every arrow in G' is in the image of f). Does this imply that $\mathrm{Path}(f)$ is also surjective? Hint: at least one of the answers to these three questions is "no".

◊

Exercise 3.3.3.7. Given a graph (V, A, src, tgt), let $i\colon A \to V \times V$ be function guaranteed by the universal property for products, as applied to $src, tgt\colon A \to V$. One might hope to summarize Condition (3.6) for graph homomorphisms by the commutativity of the single square

$$\begin{tikzcd}
A \arrow[r, "f_1"] \arrow[d, "i"'] & A' \arrow[d, "i'"] \\
V \times V \arrow[r, "f_0 \times f_0"'] & V' \times V'.
\end{tikzcd} \tag{3.8}$$

Is the commutativity of the diagram in (3.8) indeed equivalent to the commutativity of the diagrams in (3.6)? ◊

3.3.3.8 Binary relations and graphs

Definition 3.3.3.9. Let X be a set. A *binary relation on* X is a subset $R \subseteq X \times X$.

If $X = \mathbb{N}$ is the set of integers, then the usual $\leqslant$ defines a relation on X: given $(m, n) \in \mathbb{N} \times \mathbb{N}$, we put $(m, n) \in R$ iff $m \leqslant n$. As a table it might be written as to the left

$m \leqslant n$	
m	n
0	0
0	1
1	1
0	2
1	2
2	2
0	3
$\vdots$	$\vdots$

$n = 5m$	
m	n
0	0
1	5
2	10
3	15
4	20
5	25
6	30
$\vdots$	$\vdots$

$\lvert n - m\rvert \leqslant 1$	
m	n
0	0
0	1
1	0
1	1
1	2
2	1
2	2
$\vdots$	$\vdots$

(3.9)

The middle table is the relation $\{(m, n) \in \mathbb{N} \times \mathbb{N} \mid n = 5m\} \subseteq \mathbb{N} \times \mathbb{N}$ and the right-hand table is the relation $\{(m, n) \in \mathbb{N} \times \mathbb{N} \mid |n - m| \leqslant 1\} \subseteq \mathbb{N} \times \mathbb{N}$.

Exercise 3.3.3.10. A relation on $\mathbb{R}$ is a subset of $\mathbb{R} \times \mathbb{R}$, and one can indicate such a subset of the plane by shading. Choose an error bound $\epsilon > 0$ and draw the relation one might refer to as "ϵ-approximation". To say it another way, draw the relation "x is within ϵ of y". ◊

Exercise 3.3.3.11 (Binary relations to graphs). a.) If $R \subseteq S \times S$ is a binary relation, find a natural way to make a graph out of it, having vertices S.

b.) What is the set A of arrows?

c.) What are the source and target functions $src, tgt \colon A \to S$?

d.) Take the left-hand table in (3.9) and consider its first 7 rows (i.e. forget the $\vdots$). Draw the corresponding graph (do you see a tetrahedron?).

e.) Do the same for the right-hand table.

◊

Exercise 3.3.3.12 (Graphs to binary relations).

a.) If (V, A, src, tgt) is a graph, find a natural way to make a binary relation $R \subseteq V \times V$ out of it.

b.) Take the left-hand graph G from (3.7) and write out the corresponding binary relation in table form.

◊

Exercise 3.3.3.13 (Going around the loops). a.) Given a binary relation $R \subseteq S \times S$, you know from Exercise 3.3.3.11 how to construct a graph out of it, and from Exercise 3.3.3.12 how to make a new binary relation out of that. How does the resulting relation compare with the original?

b.) Given a graph (V, A, src, tgt), you know from Exercise 3.3.3.12 how to make a new binary relation out of it, and from Exercise 3.3.3.11 how to construct a new graph out of that. How does the resulting graph compare with the original?

◊

3.4 Orders

People usually think of certain sets as though they just *are* ordered, e.g. that an order is ordained by God. For example the natural numbers just *are* ordered. The letters in the alphabet just *are* ordered. But in fact we put orders on sets, and some are simply more commonly used in culture. One could order the letters in the alphabet by frequency of use and e would come before a. Given different purposes, we can put different orders on the same set. For example in Exercise 4.5.1.4 we will give a different ordering on the natural numbers that is useful in elementary number theory.

In science, we might order the set of materials in two different ways. In the first, we consider material A to be “before” material B if A is an ingredient or part of B, so water would be before concrete. But we could also order materials based on how electrically conductive they are, whereby concrete would be before water. This section is about different kinds of orders.

3.4.1 Definitions of preorder, partial order, linear order

Definition 3.4.1.1. Let S be a set and $R \subseteq S \times S$ a binary relation on S; if $(s, s') \in R$ we will write $s \leqslant s'$. Then we say that R is a *preorder* if, for all $s, s', s'' \in S$ we have

Reflexivity: $s \leqslant s$, and

Transitivity: if $s \leqslant s'$ and $s' \leqslant s''$, then $s \leqslant s''$.

We say that R is a *partial order* if it is a preorder and, in addition, for all $s, s' \in S$ we have

Antisymmetry: If $s \leqslant s'$ and $s' \leqslant s$, then $s = s'$.

We say that R is a *linear order* if it is a partial order and, in addition, for all $s, s' \in S$ we have

Comparability: Either $s \leqslant s'$ or $s' \leqslant s$.

We denote such a preorder (or partial order or linear order) by $(S, \leqslant)$.

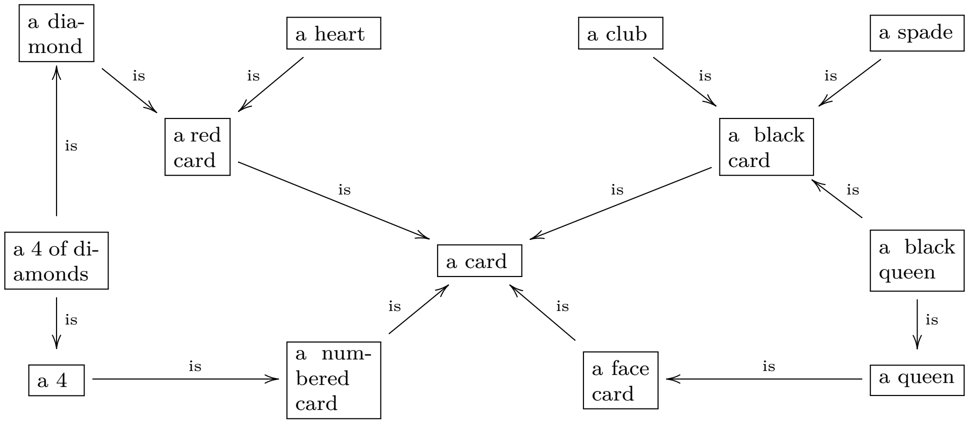
Exercise 3.4.1.2.

a.) Decide whether the table to the left in Display (3.9) constitutes a linear order.

b.) Show that neither of the other tables are even preorders.

◊

Example 3.4.1.3 (Partial order not linear order). We will draw an olog for playing cards.



(3.10)

We can put a binary relation on the set of boxes here by saying $A \leqslant B$ if there is a path $A \to B$. One can see immediately that this is a preorder because length=0 paths give reflexivity and concatenation of paths gives transitivity. To see that it is a partial order we only note that there are no loops. But this partial order is not a linear order because there is no path (in either direction) between, e.g., ⌜a 4 of diamonds⌝ and ⌜a black queen⌝, so it violates the comparability condition.

Remark 3.4.1.4. Note that olog (3.10) in Example 3.4.1.3 is a good olog in the sense that given any collection of cards (e.g. choose 45 cards at random from each of 7 decks and throw them in a pile), they can be classified according to the boxes of (3.10) such that every arrow indeed constitutes a function (which happens to be injective). For example the arrow ⌜a heart⌝ $\xrightarrow{\text{is}}$ ⌜a red card⌝ is a function from the set of chosen hearts to the set of chosen red cards.

Example 3.4.1.5 (Preorder not partial order). Every equivalence relation is a preorder but rarely are they partial orders. For example if $S = \{1, 2\}$ and we put $R = S \times S$, then this is an equivalence relation. It is a preorder but not a partial order (because $1 \leqslant 2$ and $2 \leqslant 1$, but $1 \neq 2$, so antisymmetry fails).

Application 3.4.1.6. Classically, we think of time as linearly ordered. A nice model is $(\mathbb{R}, \leqslant)$, the usual linear order on the set of real numbers. But according to the theory of relativity, there is not actually a single order to the events in the universe. Different observers correctly observe different orders on the set of events, and so in some sense on time itself.

◊◊

Example 3.4.1.7 (Finite linear orders). Let $n \in \mathbb{N}$ be a natural number. Define a linear order on the set $\{0, 1, 2, \ldots, n\}$ in the standard way. Pictorially,

$$[n] := \overset{0}{\bullet} \longrightarrow \overset{1}{\bullet} \longrightarrow \overset{2}{\bullet} \longrightarrow \cdots \longrightarrow \overset{n}{\bullet}$$

Every finite linear order, i.e. linear order on a finite set, is of the above form. That is, though the labels might change, the picture would be the same. We can make this precise when we have a notion of morphism of orders (see Definition 3.4.4.1)

Exercise 3.4.1.8. Let $S = \{1, 2, 3, 4\}$.

a.) Find a preorder $R \subseteq S \times S$ such that the set R is as small as possible. Is it a partial order? Is it a linear order?

b.) Find a preorder $R' \subseteq S \times S$ such that the set R' is as large as possible. Is it a partial order? Is it a linear order?

◊

Exercise 3.4.1.9.

a.) List all the preorder relations possible on the set $\{1, 2\}$.

b.) For any $n \in \mathbb{N}$, how many linear orders exist on the set $\{1, 2, 3, \ldots, n\}$.

c.) Does your formula work when $n = 0$?

◊

Remark 3.4.1.10. We can draw any preorder $(S, \leqslant)$ as a graph with vertices S and with an arrow $a \to b$ if $a \leqslant b$. These are precisely the graphs with the following two properties for any vertices $a, b \in S$:

1. there is at most one arrow $a \to b$, and
2. if there is a path from a to b then there is an arrow $a \to b$.

If $(S, \leqslant)$ is a partial order then the associated graph has an additional "no loops" property,

3. if $n \in \mathbb{N}$ is an integer with $n \geqslant 2$ then there are no paths of length n that start at a and end at a.

If $(S, \leqslant)$ is a linear order then there is an additional "comparability" property,

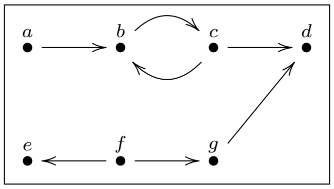
4. for any two vertices a, b there is an arrow $a \to b$ or an arrow $b \to a$.

Given a graph G, we can create a binary relation $\leqslant$ on its set S of vertices as follows. Say $a \leqslant b$ if there is a path in G from a to b. This relation will be reflexive and transitive, so it is a preorder. If the graph satisfies Property 3 then the preorder will be a partial order, and if the graph also satisfies Property 4 then the partial order will be a linear order. Thus graphs give us a nice way to visualize orders.

Slogan 3.4.1.11.

> " *A graph generates a preorder: $v \leqslant w$ if there is a path $v \to w$. This is a great way to picture a preorder.* "

Exercise 3.4.1.12. Let $G = (V, A, src, tgt)$ be the graph below.



In the corresponding pre-order which of the following are true:

a.) $a \leqslant b$?

b.) $a \leqslant c$?

c.) $c \leqslant b$?

d.) $b = c$?

e.) $e \leqslant f$?

f.) $f \leqslant d$?

◊

Exercise 3.4.1.13.

a.) Let $S = \{1, 2\}$. The subsets of S form a partial order; draw the associated graph.

b.) Repeat this for $Q = \varnothing$, $R = \{1\}$, and $T = \{1, 2, 3\}$.

c.) Do you see n-dimensional cubes?

◊

Definition 3.4.1.14. Let $(S, \leqslant)$ be a preorder. A *clique* is a subset $S' \subseteq S$ such that for each $a, b \in S'$ one has $a \leqslant b$.

Exercise 3.4.1.15. True or false: a partial order is a preorder that has no cliques. (If false, is there a "nearby" true statement?) ◊

Example 3.4.1.16. Let X be a set and $R \subseteq X \times X$ a relation. For elements $x, y \in X$ we will say there is an *R-path* from x to y if there exists a natural number $n \in \mathbb{N}$ and elements $x_0, x_1, \ldots, x_n$ such that

1. $x_0 = x$,
2. $x_n = y$, and
3. for all $i \in \mathbb{N}$, if $0 \leqslant i \leqslant n - 1$ then $(x_i, x_{i+1}) \in R$.

Let $\overline{R}$ denote the relation where $(x, y) \in \overline{R}$ if there exists an R-path from x to y. We call $\overline{R}$ the *preorder generated by R*. We note some facts about $\overline{R}$.

Containment. If $(x, y) \in R$ then $(x, y) \in \overline{R}$. That is $R \subseteq \overline{R}$.

Reflexivity . For all $x \in X$ we have $(x, x) \in \overline{R}$.

Transitivity. For all $x, y, z \in X$, if $(x, y) \in \overline{R}$ and $(y, z) \in \overline{R}$ then $(x, z) \in \overline{R}$.

To check the containment claim, just use $n = 1$ so $x_0 = x$ and $x_n = y$. To check the reflexivity claim, use $n = 0$ so $x_0 = x = y$ and condition 3 is vacuously satisfied. To check transitivitiy, suppose given R-paths $x = x_0, x_1, \ldots, x_n = y$ and $y = y_0, y_1, \ldots, y_p = z$; then $x = x_0, x_1, \ldots x_n, y_1, \ldots, y_p = z$ will be an R-path from x to z.

The point is that we can turn any relation into a preorder in a canonical way. Here is a concrete case of the above idea.

Let $X = \{a, b, c, d\}$ and suppose given the relation $\{(a, b), (b, c), (b, d), (d, c), (c, c)\}$. This is neither reflexive nor transitive, so it's not a preorder. To make it a preorder we follow the above prescription. Starting with R-paths of length $n = 0$ we put $\{(a, a), (b, b), (c, c), (d, d)\}$ into $\overline{R}$. The R-paths of length 1 add our original elements,

$\{(a,b),(b,c),(b,d),(d,c),(c,c)\}$. We don't mind redundancy (e.g. (c,c)), but from now on in this example we will only write down the new elements. The R-paths of length 2 add $\{(a,c),(a,d)\}$ to $\overline{R}$. One can check that R-paths of length 3 and above do not add anything new to $\overline{R}$, so we are done. The relation

$$\overline{R} = \{(a,a),(b,b),(c,c),(d,d),(a,b),(b,c),(b,d),(d,c),(a,c),(a,d)\}$$

is reflexive and transitive, hence a preorder.

Exercise 3.4.1.17. Let $X = \{a,b,c,d,e,f\}$ and let $R = \{(a,b),(b,c),(b,d),(d,e),(f,a)\}$.

a.) What is the preorder $\overline{R}$ generated by R?

b.) Is it a partial order?

◊

Exercise 3.4.1.18. Let X be the set of people and let $R \subseteq X \times X$ be the relation with $(x,y) \in R$ if x is the child of y. Describe the preorder generated by R. ◊

3.4.2 Meets and joins

Let X be any set. Recall from Definition 2.7.4.9 that the powerset of X, denoted $\mathbb{P}(X)$ is the set of subsets of X. There is a natural order on $\mathbb{P}(X)$ given by the subset relationship, as exemplified in Exercise 3.4.1.13. Given two elements $a,b \in \mathbb{P}(X)$ we can consider them as subsets of X and take their intersection as an element of $\mathbb{P}(X)$ which we denote $a \wedge b$. We can also consider them as subsets of X and take their union as an element of $\mathbb{P}(X)$ which we denote $a \vee b$. The intersection and union operations are generalized in the following definition.

Definition 3.4.2.1. Let $(S, \leqslant)$ be a preorder and let $s,t \in S$ be elements. A *meet of s and t* is an element $w \in S$ satisfying the following universal property:

- $w \leqslant s$ and $w \leqslant t$ and,
- for any $x \in S$, if $x \leqslant s$ and $x \leqslant t$ then $x \leqslant w$.

If w is a meet of s and t, we write $w \cong s \wedge t$.

A *join of s and t* is an element $w \in S$ satisfying the following universal property:

- $s \leqslant w$ and $t \leqslant w$ and,
- for any $x \in S$, if $s \leqslant x$ and $t \leqslant x$ then $w \leqslant x$.

If w is a join of s and t, we write $w \cong s \vee t$.

That is, the meet of s and t is the biggest thing smaller than both, i.e. a *greatest lower bound*, and the join of s and t is the smallest thing bigger than both, i.e. a *least upper bound*. Note that the meet of s and t might be s or t itself. Note that s and t may have more than one meet (or more than one join). However, any two meets of s and t must be in the same clique, by the universal property (and the same for joins).

Exercise 3.4.2.2. Consider the partial order from Example 3.4.1.3.

a.) What is the join of ⌜a diamond⌝ and ⌜a heart⌝?

b.) What is the meet of ⌜a black card⌝ and ⌜a queen⌝?

c.) What is the meet of ⌜a diamond⌝ and ⌜a card⌝?

◊

Not every two elements in a preorder need have a meet, nor need they have a join.

Exercise 3.4.2.3.

a.) If possible, find two elements in the partial order from Example 3.4.1.3 that do not have a meet. [8]

b.) If possible, find two elements that do not have a join (in that preorder).

◊

Exercise 3.4.2.4. As mentioned in the introduction to this section, the power set $S := \mathbb{P}(X)$ of any set X naturally has the structure of a partial order. Its elements $s \in S$ correspond to subsets $s \subseteq X$, and we put $s \leqslant t$ if and only if $s \subseteq t$ as subsets of X. The meet of two elements is their intersection as subsets of X, $s \wedge t = s \cap t$, and the join of two elements is their union as subsets of X, $s \vee t = s \cup t$.

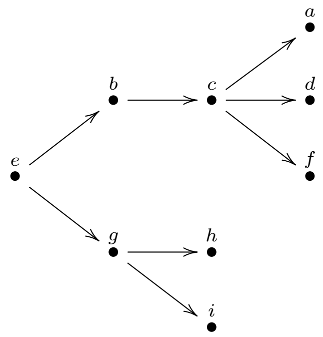
a.) Is it possible to put a monoid structure on the set S in which the multiplication formula is given by meets? If so, what would the identity element be?

b.) Is it possible to put a monoid structure on the set S in which the multiplication formula is given by joins? If so, what would the identity element be?

◊

Example 3.4.2.5 (Trees). A *tree*, i.e. a system of nodes and branches, all of which emanate from a single node called the *root*, is a partial order, but generally not a linear order. A tree $(T, \leqslant)$ can either be oriented toward the root (so the root is the largest element) or away from the root (so the root is the smallest element); let's only consider the latter.

Below is a tree, pictured as a graph. The root is labeled e.



(3.11)

In a tree, every pair of elements $s, t \in T$ has a meet $s \wedge t$ (their closest mutual ancestor). On the other hand if s and t have a join $c = s \vee t$ then either $c = s$ or $c = t$.

Exercise 3.4.2.6. Consider the tree drawn in (3.11).

a.) What is the meet $i \wedge h$?

b.) What is the meet $h \wedge b$?

c.) What is the join $b \vee a$?

d.) What is the join $b \vee g$?

◊

[8] Use the displayed preorder, not any kind of "completion of what's there".

3.4.3 Opposite order

Definition 3.4.3.1. Let $\mathcal{S} := (S, \leqslant)$ be a preorder. The *opposite preorder*, denoted $\mathcal{S}^{\text{op}}$ is the preorder $(S, \leqslant^{\text{op}})$ having the same set of elements but where $s \leqslant^{\text{op}} s'$ iff $s' \leqslant s$.

Example 3.4.3.2. Recall the preorder $\mathcal{N} := (\mathbb{N}, \texttt{divides})$ from Exercise 4.5.1.4. Then $\mathcal{N}^{\text{op}}$ is the set of natural numbers but where $m \leqslant n$ iff m is a multiple of n. So $6 \leqslant 2$ and $6 \leqslant 3$.

Exercise 3.4.3.3. Suppose that $\mathcal{S} := (S, \leqslant)$ is a preorder.

a.) If $\mathcal{S}$ is a partial order, is $\mathcal{S}^{\text{op}}$ also a partial order?

b.) If $\mathcal{S}$ is a linear order, is $\mathcal{S}^{\text{op}}$ a linear order?

◊

Exercise 3.4.3.4. Suppose that $\mathcal{S} := (S, \leqslant)$ is a preorder, and that $s_1, s_2 \in S$ have join t in $\mathcal{S}$. The preorder $\mathcal{S}^{\text{op}}$ has the same elements as $\mathcal{S}$. Is t the join of s_1 and s_2 in $\mathcal{S}^{\text{op}}$, or is it their meet, or is it not necessarily their meet nor their join? ◊

3.4.4 Morphism of orders

An order $(S, \leqslant)$, be it a preorder, a partial order, or a linear order, involves a set and a binary relations. For two orders to be comparable, their sets and their relations should be appropriately comparable.

Definition 3.4.4.1. Let $\mathcal{S} := (S, \leqslant)$ and $\mathcal{S}' := (S', \leqslant')$ be preorders (respectively partial orders or linear orders). A *morphism of preorders* (resp. *of partial orders* or *of linear orders*) *f from $\mathcal{S}$ to $\mathcal{S}'$*, denoted $f\colon \mathcal{S} \to \mathcal{S}'$, is a function $f\colon S \to S'$ such that, for every pair of elements $s_1, s_2 \in S$, if $s_1 \leqslant s_2$ then $f(s_1) \leqslant' f(s_2)$.

Example 3.4.4.2. Let X and Y be sets, let $f\colon X \to Y$ be a function. Then for every subset $X' \subseteq X$, its image $f(X') \subseteq Y$ is a subset (see Section 2.1.2). Thus we have a function $F\colon \mathbb{P}(X) \to \mathbb{P}(Y)$, given by taking images. This is a morphism of partial orders $(\mathbb{P}(X), \subseteq) \to (\mathbb{P}(Y), \subseteq)$. Indeed, if $a \subseteq b$ in $\mathbb{P}(X)$ then $f(a) \subseteq f(b)$ in $\mathbb{P}(Y)$.

Application 3.4.4.3. It's often said that "a team is only as strong as its weakest member". Is this true for materials? The hypothesis that a material is only as strong as its weakest constituent can be understood as follows.

Recall from the introduction to this section (see 3.4, page 93) that we can put several different orders on the set M of materials. One example there was the order given by constituency ($m \leqslant_C m'$ if m is an ingredient or constituent of m'). Another order is given by strength: $m \leqslant_S m'$ if m' is stronger than m (in some fixed setting).

Is it true that if material m is a constituent of material m' then the strength of m' is less than or equal to the strength of m? This is the substance of our quote above. Mathematically the question would be posed, "is there a morphism of preorders $(M, \leqslant_C) \longrightarrow (M, \leqslant_S^{\text{op}})$?"

◊◊

Exercise 3.4.4.4. Let X and Y be sets, let $f\colon X \to Y$ be a function. Then for every subset $Y' \subseteq Y$, its preimage $f^{-1}(Y') \subseteq X$ is a subset (see Definition 2.5.1.12). Thus we have a function $F\colon \mathbb{P}(Y) \to \mathbb{P}(X)$, given by taking preimages. Is it a morphism of partial orders? ◊

Example 3.4.4.5. Let S be a set. The smallest preorder structure that can be put on S is to say $a \leqslant b$ iff $a = b$. This is indeed reflexive and transitive, and it is called the *discrete preorder on S*.

The largest preorder structure that can be put on S is to say $a \leqslant b$ for all $a, b \in S$. This again is reflexive and transitive, and it is called the *indiscrete preorder on S*.

Exercise 3.4.4.6. Let S be a set and let $(T, \leqslant_T)$ be a preorder. Let $\leqslant_D$ be the discrete preorder on S. Given a morphism of preorders $(S, \leqslant_D) \to (T, \leqslant_T)$ we get a function $S \to T$.

a.) Which functions $S \to T$ arise in this way?

b.) Given a morphism of preorders $(T, \leqslant_T) \to (S, \leqslant_D)$, we get a function $T \to S$. In terms of $\leqslant_T$, which functions $T \to S$ arise in this way?

◊

Exercise 3.4.4.7. Let S be a set and let $(T, \leqslant_T)$ be a preorder. Let $\leqslant_I$ be the indiscrete preorder on S. Given a morphism of preorders $(S, \leqslant_I) \to (T, \leqslant_T)$ we get a function $S \to T$.

a.) In terms of $\leqslant_T$, which functions $S \to T$ arise in this way?

b.) Given a morphism of preorders $(T, \leqslant_T) \to (S, \leqslant_I)$, we get a function $T \to S$. In terms of $\leqslant_T$, which functions $T \to S$ arise in this way?

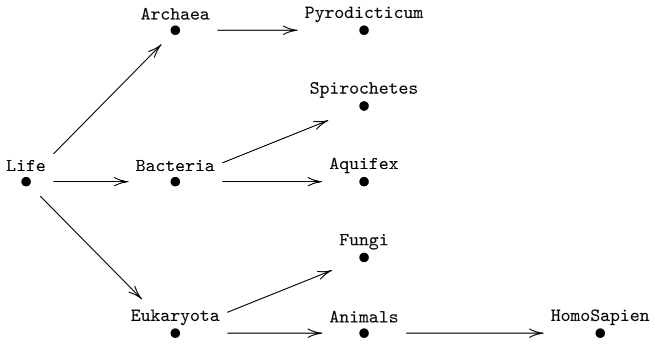
◊

3.4.5 Other applications

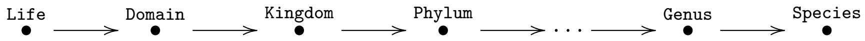
3.4.5.1 Biological classification

Biological classification is a method for dividing the set of organisms into distinct classes, called taxa. In fact, it turns out that such a classification, say a phylogenetic tree, can be understood as a partial order C on the set of taxa. The typical *ranking* of these taxa, including kingdom, phylum, etc., can be understood as morphism of orders $f \colon C \to [n]$, for some $n \in \mathbb{N}$.

For example we may have a tree (see Example 3.4.2.5) that looks like this



We also have a linear order that looks like this:



and the ranking system that puts Eukaryota at Domain and Hopo Sapien at Species is an order-preserving function from the dots upstairs to the dots downstairs; that is, it is a morphism of preorders.

Exercise 3.4.5.2. Since the phylogenetic tree is a tree, it has all meets.

a.) Determine the meet of dogs and humans.

b.) If we did not require the phylogenetic partial order to be a tree, what would it mean if two taxa (nodes in the phylogenetic partial order), say a and b, had join c with $c \neq a$ and $c \neq b$?

◊

Exercise 3.4.5.3.

a.) In your favorite scientific realm, are there any interesting classification systems that are actually orders?

b.) Choose one; what would meets and joins mean in that setting?

◊

3.4.5.4 Security

Security, say of sensitive information, is based on two things: a security clearance and "need to know." The former, security clearance might have levels like "confidential", "secret", "top secret". But maybe we can throw in "president" and some others too, like "plebe".

Exercise 3.4.5.5. Does it appear that security clearance is a preorder, a partial order, or a linear order? ◊

Need-to-know is another classification of people. For each bit of information, we do not necessarily want everyone to know about it, even everyone of the specified clearance. It is only disseminated to those that need to know.

Exercise 3.4.5.6. Let P be the set of all people and let $\overline{I}$ be the set of all pieces of information known by the government. For each subset $I \subseteq \overline{I}$, let $K(I) \subseteq P$ be the set of people that need to know every piece of information in I. Let $S = \{K(I) \mid I \subseteq \overline{I}\}$ be the set of all "need-to-know groups", with the subset relation denoted $\leqslant$.

a.) Is $(S, \leqslant)$ a preorder? If not, find a nearby preorder.

b.) If $I_1 \subseteq I_2$ do we always have $K(I_1) \subseteq K(I_2)$ or $K(I_2) \subseteq K(I_1)$ or possibly neither?

c.) Should the preorder $(S, \leqslant)$ have all meets?

d.) Should $(S, \leqslant)$ have all joins?

◊

3.4.5.7 Spaces, e.g. geography

Consider closed curves that can be drawn in the plane $\mathbb{R}^2$, e.g. circles, ellipses, and kidney-bean shaped curves. The interiors of these closed curves (not including the boundary itself) are called *basic open sets in* $\mathbb{R}^2$. The good thing about such an interior U is that any point $p \in U$ is not on the boundary, so no matter how close p is to the boundary of U, there will always be a tiny basic open set surrounding p and completely contained in U. In fact, the union of any collection of basic open sets still has this property. An *open set in* $\mathbb{R}^2$ is any subset $U \subseteq \mathbb{R}^2$ that can be formed as the union of a collection of basic open sets.

Example 3.4.5.8. Let $U = \{(x, y) \in \mathbb{R}^2 \mid x > 0\}$. To see that U is open, define the following sets: for any $a, b \in \mathbb{R}$, let $S(a, b)$ be the square parallel to the axes, with side length 1, where the upper left corner is (a, b). Let $S'(a, b)$ be the interior of $S(a, b)$. Then each $S'(a, b)$ is open, and U is the union of $S'(a, b)$ over the collection of all $a > 0$ and all b,

$$U = \bigcup_{\substack{a,\, b \,\in\, \mathbb{R},\\ a \,>\, 0}} S'(a, b).$$

The idea of open sets extends to spaces beyond $\mathbb{R}^2$. For example, on the earth one could define a basic open set to be the interior of any region one can “draw a circle around” (with a metaphorical pen), and define open sets to be unions of basic open sets.

Exercise 3.4.5.9. Let S be the set of open subsets on earth, as defined in the above paragraph.

a.) If $\leqslant$ is the subset relation, is $(S, \leqslant)$ a preorder or a partial order?

b.) Does it have meets, does it have joins?

◊

Exercise 3.4.5.10. Let S be the set of open subsets of earth as defined above. To each open subset of earth suppose we know the range of recorded temperature throughout s (i.e. the low and high throughout the region). Thus to each element $s \in S$ we assign an interval $T(s) := \{x \in \mathbb{R} \mid a \leqslant x \leqslant b\}$. If we order the set V of intervals of $\mathbb{R}$ by the subset relation, it gives a partial order on V.

a.) Does our assignment $T \colon S \to V$ amount to a morphism of orders?

b.) Does it preserve meets or joins? (Hint: it doesn’t preserve both.)

◊

Exercise 3.4.5.11.

a.) Can you think of a space relevant to your favorite area of science for which it makes sense to assign an interval of real numbers to each open set somehow, analogously to Exercise 3.4.5.10? For example for a sample of some material under stress, perhaps the strain on each open set is somehow an interval?

b.) Repeat the questions from Exercise 3.4.5.10.

◊

3.5 Databases: schemas and instances

The first three sections of this chapter were about classical objects from mathematics. The present section is about databases, which are classical objects from computer science. These are truly “categories and functors, without admitting it” (see Theorem 4.4.2.3).

3.5.1 What are databases?

Data, in particular the set of observations made during experiment, plays [9] a primary role in science of any kind. To be useful data must be organized, often in a row-and-column display called a table. Columns existing in different tables can refer to the same data.

A database is a collection of tables, each table T of which consists of a set of columns and a set of rows. We roughly explain the role of tables, columns, and rows as follows. The existence of table T suggests the existence of a fixed methodology for observing objects or events of a certain type. Each column c in T prescribes a single kind or method of observation, so that the datum inhabiting any cell in column c refers to an observation of that kind. Each row r in T has a fixed sourcing event or object, which can be observed using the methods prescribed by the columns. The cell (r, c) refers to the observation of kind c made on event r. All of the rows in T should refer to uniquely identifiable objects or events of a single type, and the name of the table T should refer to that type.

Example 3.5.1.1. When graphene is strained (lengthened by a factor of $x \geqslant 1$), it becomes stressed (carries a force in the direction of the lengthening). The following is a made-up set of data.

Graphene sample			
ID	**Source**	**Stress**	**Strain**
A118-1	C Smkt	0	0
A118-2	C Smkt	0.02	20
A118-3	C Smkt	0.05	40
A118-4	AC	0.04	37
A118-5	AC	0.1	80
A118-6	C Plat	0.1	82

Supplier		
ID	**Full name**	**Phone**
C Smkt	Carbon Supermarket	(541)781-6611
AC	Advanced Chemical	(410) 693-0818
C Plat	Carbon Platform	(510) 719-2857
McD	McDonard's Burgers	(617) 244-4400
APP	Acme Pen and Paper	(617) 823-5603

(3.12)

In the first table, titled "Graphene sample", the rows refer to graphene samples, and the table is so named. Each graphene sample can be observed according to the source supplier from which it came, the strain that it was subjected to, and the stress that it carried. These observations are the columns. In the second table, the rows refer to suppliers of various things, and the table is so named. Each supplier can be observed according to its full name and its phone number; these are the columns.

In the left-hand table it appears either that each graphene sample was used only once, or that the person recording the data did not keep track of which samples were reused. If such details become important later, the lab may want to change the layout of the first table by adding on the appropriate column. This can be accomplished using morphisms of schemas, which will be discussed in Section 4.4.1.

[9] The word data is generally considered to be the plural form of the word datum. However, individual datum elements are only useful when they are organized into structures (e.g. if one were to shuffle the cells in a spreadsheet, most would consider the data to be destroyed). It is the whole organized structure that really houses the information; the data must be in formation in order to be useful. Thus I will use the word *data* as a collective noun (akin to the word "sand"); it bridges the divide between the *individual datum elements* (akin to the grains of sand) and the *data set* (akin to a sand pile). In particular, I will often use the word data as a singular noun.

3.5.1.2 Primary keys, foreign keys, and data columns

There is a bit more structure in the above tables (Example 3.12) then may first meet the eye. Each table has a *primary ID column*, found on the left, as well as some *data columns* and some *foreign key columns*. The primary key column is tasked with uniquely identifying different rows. Each data column houses elementary data of a certain sort. Perhaps most interesting from a structural point of view are the foreign key columns, because they link one table to another, creating a connection pattern between tables. Each foreign key column houses data that needs to be further unpacked. It thus refers us to another *foreign* table, in particular the primary ID column of that table. In Example 3.12 the `Source` column was a foreign key to the `Supplier` table.

Here is another example, lifted from [Sp2].

Example 3.5.1.3. Consider the bookkeeping necessary to run a department store. We keep track of a set of employees and a set of departments. For each employee e, we keep track of

E.1 the **first** name of e, which is a `FirstNameString`,

E.2 the **last** name of e, which is a `LastNameString`,

E.3 the **manager** of e, which is an `Employee`, and

E.4 the department that e **works in**, which is a `Department`.

For each department d, we keep track of

D.1 the **name** of d, which is a `DepartmentNameString`, and

D.2 the **secretary** of d, which is an `Employee`.

Above we can suppose that E.1, E.2, and D.1 are data columns (referring to names of various sorts), and E.3, E.4, and D.2 are foreign key columns (referring to managers, secretaries, etc.).

Display (3.13) shows how such a database might look at a particular moment in time.

`Employee`				
ID	**first**	**last**	**manager**	**worksIn**
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Emmy	Noether	103	q10

`Department`		
ID	**name**	**secretary**
q10	Sales	101
x02	Production	102

(3.13)

3.5.1.4 Business rules

Looking at the tables from Example 3.5.1.3, one may notice a few patterns. First, every employee works in the same department as his or manager. Second, every department's secretary works in that department. Perhaps the business counts on these rules for the way it structures itself. In that case the database should enforce those rules, i.e. it should check that whenever the data is updated, it conforms to the rules:

Rule 1 For every employee e, the **manager** of e **works in** the same department that e **works in**.

Rule 2 For every department d, the **secretary** of d **works in** department d.

(3.14)

Together, the statements E.1, E.2, E.3, E.4, D.1, and D.2 from Example 3.5.1.3 and Rule 1 and Rule 2, constitute what we will call the *schema* of the database. We will formalize this idea in Section 3.5.2.

3.5.1.5 Data columns as foreign keys

To make everything consistent, we could even say that data columns are specific kinds of foreign keys. That is, each data column constitutes a foreign key to some non-branching *leaf table*, which has no additional data.

Example 3.5.1.6. Consider again Example 3.5.1.3. Note that first names and last names had a particular type, which we all but ignored above. We could cease to ignore them by adding three tables, as follows.

FirstNameString
ID
Alan
Alice
Bertrand
Carl
David
Emmy
⋮

LastNameString
ID
Arden
Hilbert
Jones
Noether
Russell
⋮

DepartmentNameString
ID
Marketing
Production
Sales
⋮

(3.15)

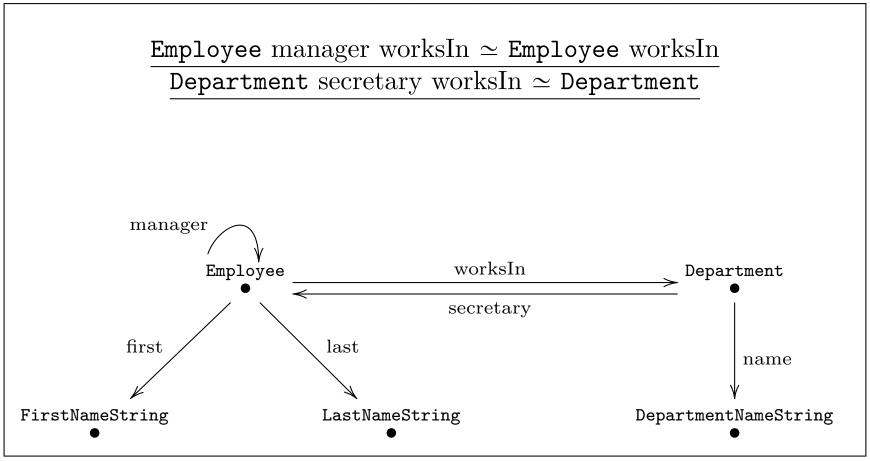
In combination, Displays (3.13) and (3.15) form a collection of tables with the property that every column is either a primary key or a foreign key. The notion of data column is now subsumed under the notion of foreign key column. Everything is either a primary key (one per table, labeled ID) or a foreign key column (everything else).

3.5.2 Schemas

The above section may all seem intuitive or reasonable in some ways, but also a bit difficult to fully grasp, perhaps. It would be nice to summarize what is happening in a picture. Such a picture, which will basically be a graph, should capture the *conceptual layout* to which the data conforms, without yet being concerned with the individual data that may populate the tables in this instant. We proceed at first by example, giving the precise definition in Definition 3.5.2.6.

Example 3.5.2.1. In Examples 3.5.1.3 and 3.5.1.6, the conceptual layout for a department store was given, and some example tables were shown. We were instructed to keep track of employees, departments, and six types of data (E.1, E.2, E.3, E.4, D.1, and D.2), and we were instructed to follow two rules (Rule 1, Rule 2). All of this is summarized in the following picture:

$\mathcal{C}$:= Schema for tables (3.13) and (3.15) conforming to (3.14)



(3.16)

The five tables from (3.13) and (3.15) are seen as five vertices; this is also the number of primary ID columns. The six foreign key columns from (3.13) and (3.15) are seen as six arrows; each points from a table to a foreign table. The two rules from (3.14) are seen as statements at the top of Display (3.16).We will explain path equivalences in Definition 3.5.2.3.

Exercise 3.5.2.2. Come up with a schema (consisting of dots and arrows) describing the conceptual layout of information presented in Example 3.5.1.1. ◊

In order to define schemas, we must first define the notion of schematic equivalence relation, which is to hold on the set of paths of a graph G (see Section 3.3.2). Such an equivalence relation (in addition to being reflexive, symmetric, and transitive) has two sorts of additional properties: equivalent paths must have the same source and target, and the composition of equivalent paths with other equivalent paths must yield equivalent paths. Formally we have Definition 3.5.2.3.

Definition 3.5.2.3.

Let $G = (V, A, src, tgt)$ be a graph, and let Path_G denote the set of paths in G (see Definition 3.3.2.1). A *path equivalence declaration* (or *PED*) is an expression of the form $p \simeq q$ where $p, q \in \mathrm{Path}_G$ have the same source and target, $src(p) = src(q)$ and $tgt(p) = tgt(q)$.

A *congruence* on G is a relation $\simeq$ on Path_G that has the following properties:

1. The relation $\simeq$ is an equivalence relation.

2. If $p \simeq q$ then $src(p) = src(q)$.

3. If $p \simeq q$ then $tgt(p) = tgt(q)$.

4. Suppose $p, q\colon b \to c$ are paths, and $m\colon a \to b$ is an arrow. If $p \simeq q$ then $mp \simeq mq$.

5. Suppose $p, q\colon a \to b$ are paths, and $n\colon b \to c$ is an arrow. If $p \simeq q$ then $pn \simeq qn$.

Any set of path equivalence declarations (PEDs) generates a congruence. We tend to elide the difference between a congruence and the set of PEDs that generates it.

Exercise 3.5.2.4. Consider the graph shown in (3.16), and the two declarations shown at the top. They generate a congruence.

a.) Is it true that the following PED is an element of this congruence?

$$\texttt{Employee}\text{ manager manager worksIn} \overset{?}{\simeq} \texttt{Employee}\text{ worksIn}$$

b.) What about this one?

$$\texttt{Employee}\text{ worksIn secretary} \overset{?}{\simeq} \texttt{Employee}$$

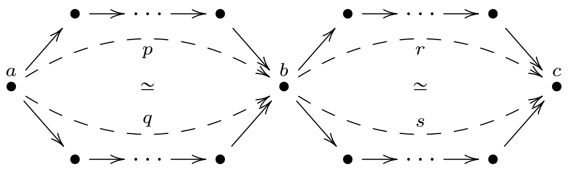
c.) What about this one?

$$\texttt{Department}\text{ secretary manager worksIn name} \overset{?}{\simeq} \texttt{Department}\text{ name}$$

◊

Lemma 3.5.2.5. *Suppose that G is a graph and $\simeq$ is a congruence on G. Suppose $p \simeq q\colon a \to b$ and $r \simeq s\colon b \to c$. Then $pr \simeq qs$.*

Proof. The picture to have in mind is this:



Applying condition (3) from Definition 3.5.2.3 to each arrow in path p, it follows by induction that $pr \simeq ps$. Applying condition (4) to each arrow in path s, it follows similarly that $ps \simeq qs$. Because $\simeq$ is an equivalence relation, it follows that $pr \simeq qs$.

□

Definition 3.5.2.6. A *database schema* (or simply *schema*) $\mathcal{C}$ consists of a pair $\mathcal{C} := (G, \simeq)$ where G is a graph and $\simeq$ is a congruence on G.

Example 3.5.2.7. The picture drawn in (3.16) has the makings of a schema. Pictured is a graph with two PEDs; these generate a congruence, as discussed in Exercise 3.5.2.4.

A schema can be converted into a system of tables each with a primary key and some number of foreign keys referring to other tables, as discussed in Section 3.5.1. Definition 3.5.2.6 gives a precise conceptual understanding of what a schema is, and the following rules describe how to convert such a thing into a table layout.

Rules of good practice 3.5.2.8. Converting a schema $\mathcal{C} = (G, \simeq)$ into a table layout should be done as follows:

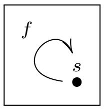
(i) There should be a table for every vertex in G and if the vertex is named, the table should have that name;

(ii) Each table should have a left-most column called ID, set apart from the other columns by a double vertical line; and

(iii) To each arrow a in G having source vertex $s := src(a)$ and target vertex $t := tgt(a)$, there should be a foreign key column a in table s, referring to table t; if the arrow a is named, column a should have that name.

Example 3.5.2.9 (Discrete dynamical system). Consider the schema

$\mathcal{L}oop :=$

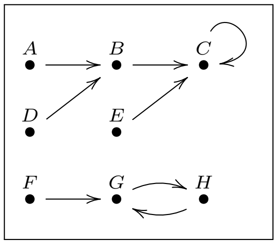


(3.17)

in which the congruence is trivial (i.e. generated by the empty set of PEDs.) This schema is quite interesting. It encodes a set s and a function $f\colon s \to s$. Such a thing is called a *discrete dynamical system*. One imagines s as the set of states and, for any state $x \in s$, a notion of "next state" $f(x) \in s$. For example

s	
ID	**f**
A	B
B	C
C	C
D	B
E	C
F	G
G	H
H	G

...pictured...

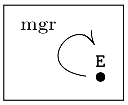


(3.18)

Application 3.5.2.10. Imagine a quantum-time universe in which there are discrete time steps. We model it as a discrete dynamical system, i.e. a table of the form (3.18). For every possible state of the universe we include a row in the table. The state in the next instant is recorded in the second column.

◊◊

Example 3.5.2.11 (Finite hierarchy). The schema $\mathcal{L}oop$ can also be used to encode hierarchies, such as the manager relation from Examples 3.5.1.3 and 3.5.2.1,



One problem with this, however, is if a schema has even one loop, then it can have infinitely many paths (corresponding, e.g. to an employees manager's manager's manager's ... manager).

Sometimes we know that in a given company that process eventually ends, a famous example being that at Ben and Jerry's ice cream, there were only seven levels. In that case we know that an employee's 8th level manager is equal to his or her 7th level manager. This can be encoded by the PED

$$\texttt{E}\text{ mgr mgr mgr mgr mgr mgr mgr mgr} \simeq \texttt{E}\text{ mgr mgr mgr mgr mgr mgr mgr}$$

or more concisely, $\text{mgr}^8 = \text{mgr}^7$.

Exercise 3.5.2.12. Is there any nontrivial PED on $\mathcal{L}oop$ that holds for the data in Example 3.5.2.9? If so, what is it and how many equivalence classes of paths in $\mathcal{L}oop$ are there after you impose that relation? ◊

Exercise 3.5.2.13. Let P be a chess-playing program. Given any position (including the history of the game and choice of whose turn it is), P will make a move.

a.) Is this an example of a discrete dynamical system?

b.) How do the rules for ending the game in a win or draw play out in this model? (Look up online how chess games end if you don't know.)

◊

3.5.2.14 Ologging schemas

It should be clear that a database schema is nothing but an olog in disguise. The difference is basically the readability requirements for ologs. There is an important new addition in this section, namely that we can fill out an olog with data. Conversely, we have seen that databases are not any harder to understand than ologs are.

Example 3.5.2.15. Consider the olog

$$\begin{tikzcd}\boxed{\text{a moon}} \arrow[r, "\text{orbits}"] & \boxed{\text{a planet}}\end{tikzcd} \tag{3.19}$$

We can document some instances of this relationship using the following tables:

orbits	
a moon	**a planet**
The Moon	Earth
Phobos	Mars
Deimos	Mars
Ganymede	Jupiter
Titan	Saturn

(3.20)

Clearly, this table of instances can be updated as more moons are discovered by the author (be it by telescope, conversation, or research).

Exercise 3.5.2.16. In fact, Example 3.5.2.15 did not follow Rules 3.5.2.8. Strictly following those rules, copy over the data from (3.20) into tables that are in accordance with schema (3.19). ◊

Exercise 3.5.2.17.

a.) Write down a schema, in terms of the boxes ⌜a thing I own⌝ and ⌜a place⌝ and one additional arrow, that might help one remember where they decided to put "random" things.

b.) What is a good label for the arrow?

c.) Fill in some rows of the corresponding set of tables for your own case.

◊

Exercise 3.5.2.18. Consider the olog

$$\begin{tikzcd}\overset{C}{\boxed{\text{a child}}} \arrow[r, "\text{has}"] & \overset{F}{\boxed{\text{a father}}} \arrow[l, bend right, "\text{has as tallest}"'] \arrow[l, bend left, "\text{has as first}"]\end{tikzcd}$$

a.) What path equivalence declarations would be appropriate for this olog? You can use $f\colon F \to C$, $t\colon F \to C$, and $h\colon C \to F$ if you prefer.

b.) How many PEDs are in the congruence?

$\Diamond$

3.5.3 Instances

Given a database schema $(G, \simeq)$, an instance of it is just a bunch of tables whose data conform to the specified layout. These can be seen throughout the previous section, most explicitly in the relationship between schema (3.16) and tables (3.13) and (3.15), and between schema (3.17) and table (3.18). Below is the mathematical definition.

Definition 3.5.3.1. Let $\mathcal{C} = (G, \simeq)$ where $G = (V, A, src, tgt)$. An *instance on* $\mathcal{C}$, denoted $(\text{PK}, \text{FK})\colon \mathcal{C} \to \textbf{Set}$, is defined as follows: One announces some constituents (A. primary ID part, B. foreign key part) and asserts that they conform to a law (1. preservation of congruence). Specifically, one announces

A. a function $\text{PK}\colon V \to \textbf{Set}$; i.e. to each vertex $v \in V$ one provides a set $\text{PK}(v)$;[10] and

B. for every arrow $a \in A$ with $v = src(a)$ and $w = tgt(a)$, a function $\text{FK}(a)\colon \text{PK}(v) \to \text{PK}(w)$. [11]

One asserts that the following law holds for any vertices v, w and paths $p = va_1a_2\ldots a_m$ and $q = va'_1a'_2\ldots a'_n$ from v to w:

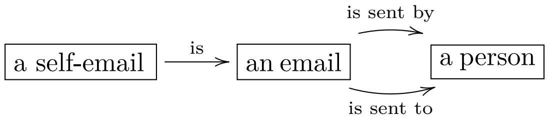
1. If $p \simeq q$ then for all $x \in \text{PK}(v)$, we have

$$\text{FK}(a_m) \circ \cdots \circ \text{FK}(a_2) \circ \text{FK}(a_1)(x) = \text{FK}(a'_n) \circ \cdots \circ \text{FK}(a'_2) \circ \text{FK}(a'_1)(x)$$

in $\text{PK}(w)$.

Exercise 3.5.3.2. Consider the olog pictured below:

$\mathcal{C} :=$



Given x, a self-email, consider the following.
We know that x is a self-email, which is an email, which is sent by a person that we'll call $P(x)$.
We also know that x is a self-email, which is an email, which is sent to a person that we'll call $Q(x)$.
Fact: whenever x is a self-email, we will have $P(x) = Q(x)$

[10] The elements of $\text{PK}(v)$ will be listed as the rows of table v, or more precisely as the leftmost cells of these rows.

[11] The arrow a will correspond to a column, and to each row $r \in \text{PK}(v)$ the (r, a) cell will contain the datum $\text{FK}(a)(r)$.

`a self-email`	
ID	**is**
SEm1207	Em1207
SEm1210	Em1210
SEm1211	Em1211

`an email`		
ID	**is sent by**	**is sent to**
Em1206	Bob	Sue
Em1207	Carl	Carl
Em1208	Sue	Martha
Em1209	Chris	Bob
Em1210	Chris	Chris
Em1211	Julia	Julia
Em1212	Martha	Chris

`a person`
ID
Bob
Carl
Chris
Julia
Martha
Sue

(3.21)

a.) What is the set $\text{PK}(\ulcorner\text{an email}\urcorner)$?

b.) What is the set $\text{PK}(\ulcorner\text{a person}\urcorner)$?

c.) What is the function $\text{FK}(\text{is sent by})\colon \text{PK}(\ulcorner\text{an email}\urcorner) \to \text{PK}(\ulcorner\text{a person}\urcorner)$?

d.) Interpret the sentences at the bottom of $\mathcal{C}$ as the Englishification of a simple path equivalence declaration. Is it satisfied by the instance (3.21); that is, does law 1. from Definition 3.5.3.1 hold?

$\Diamond$

Example 3.5.3.3 (Monoid action table). In Example 3.1.2.9, we saw how a monoid $\mathcal{M}$ could be captured as an olog with only one object. As a database schema, this means there is only one table. Every generator of $\mathcal{M}$ would be a column of the table. The notion of database instance for such a schema is precisely the notion of action table from Section 3.1.3. Note that a monoid can act on itself, in which case this action table is the monoid's multiplication table as in Example 3.1.3.2, but it can also act on any other set as in Example 3.1.3.1. If $\mathcal{M}$ acts on a set S, then the set of rows in the action table will be S.

Exercise 3.5.3.4. Draw (as a graph) the schema for which Table 3.2 is an instance. $\Diamond$

Exercise 3.5.3.5. Suppose that $\mathcal{M}$ is a monoid and some instance of it is written out in table form. It's possible that $\mathcal{M}$ is a group. What evidence in an instance table for $\mathcal{M}$ might suggest that $\mathcal{M}$ is a group? $\Diamond$

3.5.3.6 Paths through a database

Let $\mathcal{C} := (G, \simeq)$ be a schema and let $(\text{PK}, \text{FK})\colon \mathcal{C} \to \textbf{Set}$ be an instance on $\mathcal{C}$. Then for every arrow $a\colon v \to w$ in G we get a function $\text{FK}(a)\colon \text{PK}(v) \to \text{PK}(w)$. Functions can be composed, so in fact for every path through G we get a function. Namely, if $p = v_0 a_1, a_2, \ldots, a_n$ is a path from v_0 to v_n then the instance provides a function

$$\text{FK}(p) := \text{FK}(a_n) \circ \cdots \text{FK}(a_2) \circ \text{FK}(a_1)\colon \text{PK}(v_0) \to \text{PK}(v_n),$$

which first made an appearance as part of Law 1 in Definition 3.5.3.1.

Example 3.5.3.7. Consider the department store schema from Example 3.5.2.1, and in (3.16) the path [worksIn, secretary, last] which points from `Employee` to `LastNameString`. The instance will let us interpret this path as a function from the set of employees to the set of last names; this could be a useful function to have around. The instance from (3.13) would yield the following function

Employee	
ID	**Secr. name**
101	Hilbert
102	Russell
103	Hilbert

Exercise 3.5.3.8. Consider the path $p := [f, f]$ on the $\mathcal{L}oop$ schema from (3.17). Using the instance from (3.18), where $\text{PK}(s) = \{A, B, C, D, E, F, G, H\}$, interpret p as a function $\text{PK}(s) \to \text{PK}(s)$, and write this as a 2-column table, as above in Example 3.5.3.7. ◊

Exercise 3.5.3.9.

a.) Given an instance (PK, FK) on a schema $\mathcal{C}$, and given a trivial path p (i.e. p has length 0; it starts at some vertex but doesn't go anywhere), what function does p yield?

b.) What are the domain and codomain of p?

◊

Chapter 4

Basic category theory

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

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

4.1 Categories and Functors

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

4.1.1 Categories

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

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

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

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

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

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

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

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

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

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

called *the composition formula*.

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

One asserts that the following law holds:

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

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

and;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

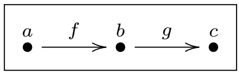
Exercise 4.1.1.6 (The category **Grp** of groups). Suppose we set out to define a category **Grp**, having groups as objects and group homomorphisms as morphisms, see Definition 3.2.1.16. Show (to the level of detail of Example 4.1.1.5) that the rest of the conditions for **Grp** to be a category are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

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

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

Let G be the following graph,

$G =$



.

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

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

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

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

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

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

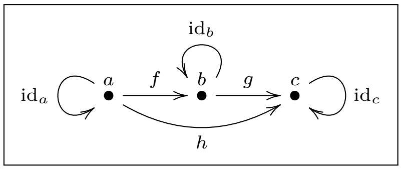
What about a composition formula? We need a function $\mathrm{Hom}_{\mathcal{G}}(a,b) \times \mathrm{Hom}_{\mathcal{G}}(b,c) \to \mathrm{Hom}_{\mathcal{G}}(a,c)$, but the domain is nonempty and the codomain is empty; there is no such function.

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

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

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

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

$G =$ 

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.1.1.16 Isomorphisms

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

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

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

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

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

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

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

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

◊

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

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

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

◊

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

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

4.1.1.22 Another viewpoint on categories

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

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

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

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

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

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

[4] See Remark 4.1.1.2.

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

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

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

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

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

◊

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

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

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

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

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

4.1.2 Functors

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

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

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

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

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

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

One asserts that the following laws hold:

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

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

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

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

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

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

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

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

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

Slogan 4.1.2.4.

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

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

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

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

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

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

◊◊

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

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

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

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

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

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

a.) Is there an "obvious" functor **FLin** → **PrO**?

b.) Is there an "obvious" functor **PrO** → **FLin**?

◊

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

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

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

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

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

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

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

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

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

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

surjective)?

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

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

◊

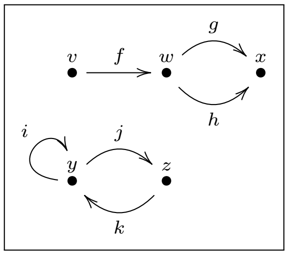
Remark 4.1.2.10. There is a functor $W\colon \mathbf{PrO} \to \mathbf{Set}$ sending $(X, \leqslant)$ to X. There is a functor $T\colon \mathbf{Grph} \to \mathbf{Set}$ sending (V, A, src, tgt) to V. When we understand the category of categories (Section 4.1.2.27), it will be clear that Proposition 4.1.2.8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

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

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

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

$G :=$



◊

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

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

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

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

◊

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

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

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

◊

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

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

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

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

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

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

Exercise 4.1.2.17.

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

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

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

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

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

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

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

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

◊◊

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

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

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

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

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

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

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

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

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

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

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

◊

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

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

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

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

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

Exercise 4.1.2.23.

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

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

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

◊

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

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

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

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

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

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

4.1.2.27 The category of categories

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

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

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

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

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

□

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

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

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

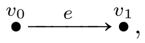
Here, the objects of the category $F(G)$ are the vertices of G. For any two vertices v, v' the hom-set $\mathrm{Hom}_{F(G)}(v,v')$ is the set of paths in G from v to v'. The identity elements are given by the trivial paths, and the composition formula is given by concatenation of paths.

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

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

called *the free category* functor.

Exercise 4.1.2.31*.* Let G be the graph depicted



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

a.) What are its objects?

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

◊

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

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

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

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

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

◊

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

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

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

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

a.) List all the morphisms in D_4.

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

◊

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

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

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

4.2 Categories and functors commonly arising in mathematics

4.2.1 Monoids, groups, preorders, and graphs

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

4.2.1.1 Monoids as categories

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

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

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

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

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

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

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

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

Slogan 4.2.1.2.

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

We formalize this as the following theorem.

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

- *for every monoid $\mathcal{M} \in \text{Ob}(\textbf{Mon})$, the category $i(\mathcal{M}) \in \text{Ob}(\textbf{Cat})$ itself has exactly one object,*

$$|\text{Ob}(i(\mathcal{M}))| = 1$$

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

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

  induced by the functor i, is a bijection.

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

□

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

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

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

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

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

4.2.1.4 Groups as categories

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

Slogan 4.2.1.5.

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

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

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

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

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

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

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

4.2.1.7 Monoid and group stationed at each object in a category

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

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

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

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

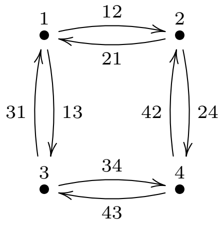
a.) What is the automorphism group of S in **Set**, and how many elements does this group have?

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

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

◊

Exercise 4.2.1.11. Consider the graph G depicted below.



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

4.2.1.12 Preorders as categories

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

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

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

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

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

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

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

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

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

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

and we have

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

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

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

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

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

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

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

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

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

□

Slogan 4.2.1.18.

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

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

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

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

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

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

◊

4.2.1.20 Graphs as functors

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

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

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

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

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

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

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

with the following composition formula:

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

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

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

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

Slogan 4.2.1.23.

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

Exercise 4.2.1.24. Which of the following graphs are symmetric:

a.) The graph G from (3.4)?

b.) The graph G from Exercise 3.3.1.10?

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

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

e.) The graph G from Exercise 4.2.1.11?

◊

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

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

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

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

◊

4.2.2 Database schemas present categories

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

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

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

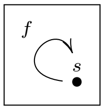
Slogan 4.2.2.1.

> " *A database schema is a category presentation.* "

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

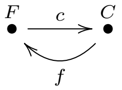
Example 4.2.2.2. The schema $\mathcal{L}oop$, depicted below, has no path equivalence declarations. As a graph it has one vertex and one arrow.

$\mathcal{L}oop :=$



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

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



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

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

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

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

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

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

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

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

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

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

$\Diamond$

4.2.3 Spaces

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

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

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

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

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

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

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

Exercise 4.2.3.2.

a.) Explain how "looking at points" gives a functor **Top** → **Set**.

b.) Does "looking at open sets" give a functor **Top** → **PrO**?

◊

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

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

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

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

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

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

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

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

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

◊

4.2.3.6 Groupoids

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

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

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

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

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

◊◊

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

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

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

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

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

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

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

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

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

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

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

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

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

◊

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

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

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

4.2.4 Logic, set theory, and computer science

4.2.4.1 The category of propositions

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

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

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

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

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

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

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

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

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

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

♢

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

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

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

♢

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

4.2.4.5 A categorical characterization of Set

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

4.2.4.6 Categories in computer science

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

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

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

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

4.2.5 Categories applied in science

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

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

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

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

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

4.3 Natural transformations

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

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

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

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

4.3.1 Definition and examples

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

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

Slogan 4.3.1.1.

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

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

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

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

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

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

One asserts that the following law holds:

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

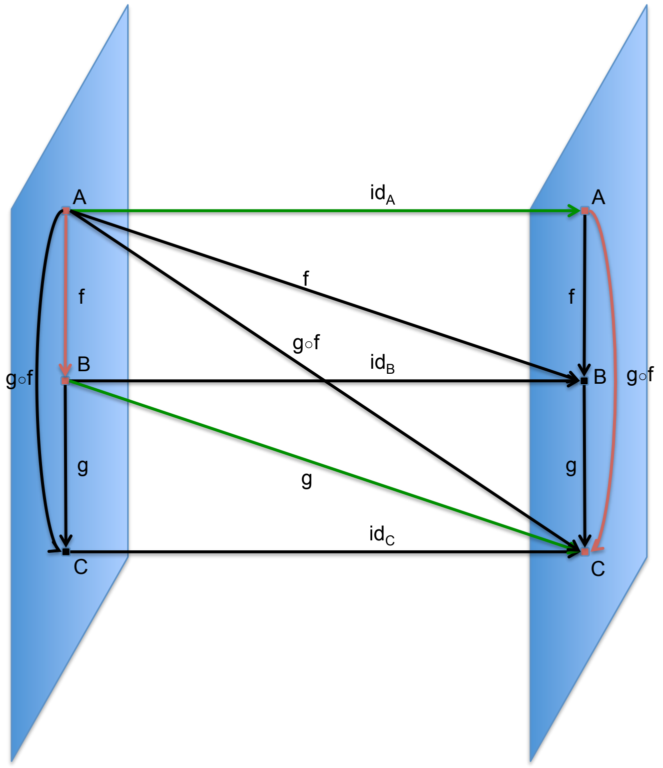
square for h, must commute:

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

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

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

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



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

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

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

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

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

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

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

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

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

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

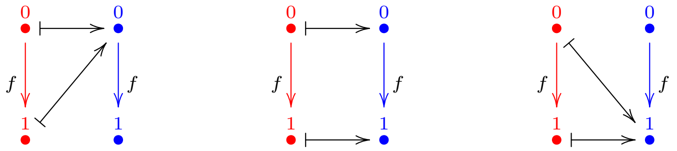
commutes. But this is clear because functors preserve identities.

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

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

□

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



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

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

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

This commutes, so α is indeed a natural transformation.

Exercise 4.3.1.6. With notation as in Example 4.3.1.5,

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

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

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

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

$\Diamond$

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

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

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

$\Diamond$

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

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

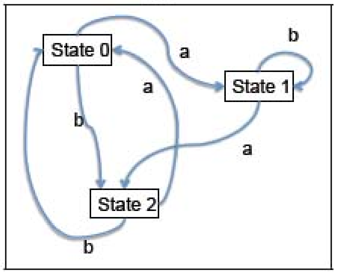
b.) Let $\mathcal{D} = \textbf{Set}$ and let $d = \{☺\}$. If $\mathcal{C} = [1]$ is the linear order of length 1, and $F\colon \mathcal{C} \to \textbf{Set}$ is any functor, what does it mean to give a natural transformation $\{d\}^{\mathcal{C}} \to F$?

◊

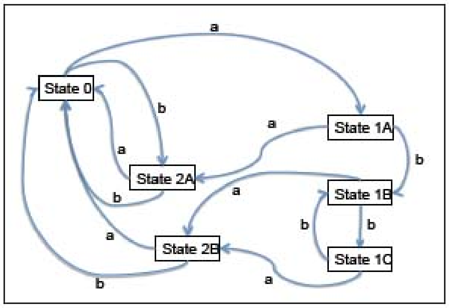
Application 4.3.1.9. In Figure 3.1 we drew a finite state machine on alphabet $\Sigma = \{a, b\}$, and in Example 3.1.3.1 we showed the associated action table. It will be reproduced below. Imagine this was your model for understanding the behavior of some system when acted on by commands a and b. And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

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

X:=



Y:=



The action tables for these two machines are:

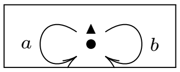
Original model X		
ID	**a**	**b**
State 0	State 1	State 2
State 1	State 2	State 1
State 2	State 0	State 0

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

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

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

$\mathcal{M} =$



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

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

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

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

Naturality square for $a\colon \blacktriangle \to \blacktriangle$

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

(4.11)

Naturality square for $b\colon \blacktriangle \to \blacktriangle$

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

(4.12)

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

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

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

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

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

◊◊

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

a.) What might they mean?

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

◊

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

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

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

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

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

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

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

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

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

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

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

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

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

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

b.) What are its components?

◊

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

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

b.) What are its components?

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

◊

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

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

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

4.3.2 Vertical and horizontal composition

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

$$\begin{tikzcd}
\mathcal{C} \arrow[r, bend left=60, "F"] \arrow[r, "G" description] \arrow[r, bend right=60, "H"'] \arrow[r, phantom, bend left=30, "\alpha\Downarrow"] \arrow[r, phantom, bend right=30, "\beta\Downarrow"] & \mathcal{D}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{C} \arrow[r, bend left=40, "F_1"] \arrow[r, bend right=40, "G_1"'] \arrow[r, phantom, "\gamma_1\Downarrow"] & \mathcal{D} \arrow[r, bend left=40, "F_2"] \arrow[r, bend right=40, "G_2"'] \arrow[r, phantom, "\gamma_2\Downarrow"] & \mathcal{E}
\end{tikzcd}$$

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

4.3.2.1 Vertical composition of natural transformations

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

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

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

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

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

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

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

□

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

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

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

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

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

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

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

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

$\lozenge$

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

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

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

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

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

4.3.2.11 Natural isomorphisms

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

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

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

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

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

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

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

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

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

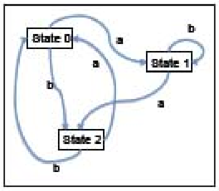
□

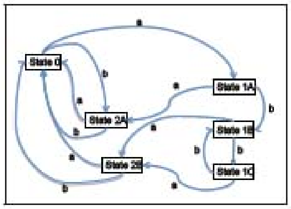
Exercise 4.3.2.13. Recall from Application 4.3.1.9 that a finite state machine on alphabet Σ can be understood as a functor $\mathcal{M} \to \mathbf{Set}$, where $\mathcal{M} = \mathrm{List}(\Sigma)$ is the free monoid generated by Σ. In that example we also discussed how natural transformations provide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

4.3.2.14 Horizontal composition of natural transformations

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

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





Original model $X\colon \mathcal{M} \to \mathbf{Set}$

ID	**a**	**b**
State 0	State 1	State 2
State 1	State 2	State 1
State 2	State 0	State 0

Proposed model $Y\colon \mathcal{M} \to \mathbf{Set}$

ID	**a**	**b**
State 0	State 1A	State 2A
State 1A	State 2A	State 1B
State 1B	State 2B	State 1C
State 1C	State 2B	State 1B
State 2A	State 0	State 0
State 2B	State 0	State 0

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

$$\begin{tikzcd}
\mathcal{M}' \arrow[r, "F"] & \mathcal{M} \arrow[r, bend left, "Y"{name=U}] \arrow[r, bend right, "X"'{name=D}] & \mathbf{Set} \arrow[from=U, to=D, phantom, "\alpha\Downarrow"]
\end{tikzcd}$$

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

$X \circ F$

ID	**m**	**n**	**p**
State 0	State 1	State 2	State 1
State 1	State 2	State 1	State 0
State 2	State 0	State 0	State 2

$Y \circ F$

ID	**m**	**n**	**p**
State 0	State 1A	State 2A	State 1A
State 1A	State 2A	State 1B	State 0
State 1B	State 2B	State 1C	State 0
State 1C	State 2B	State 1B	State 0
State 2A	State 0	State 0	State 2A
State 2B	State 0	State 0	State 2A

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Theorem 4.3.2.20.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, bend left=60, "F_1"] \arrow[r, "F_2" description] \arrow[r, bend right=60, "F_3"'] \arrow[r, phantom, bend left=30, "\alpha_1\Downarrow"] \arrow[r, phantom, bend right=30, "\alpha_2\Downarrow"] & \mathcal{D} \arrow[r, bend left=60, "G_1"] \arrow[r, "G_2" description] \arrow[r, bend right=60, "G_3"'] \arrow[r, phantom, bend left=30, "\beta_1\Downarrow"] \arrow[r, phantom, bend right=30, "\beta_2\Downarrow"] & \mathcal{E}
\end{tikzcd}$$

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

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

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

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

□

4.3.3 The category of instances on a database schema

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

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

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

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

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

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

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

Grapes (I)
ID
Grape 1
Grape 3
Grape 4

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

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

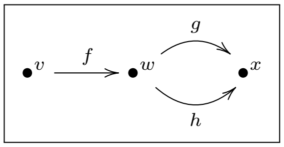
Exercise 4.3.3.4. Recall the notion of set-indexed sets from Definition 2.7.6.12. Let A be a set, and come up with a schema $\mathcal{A}$ such that instances on $\mathcal{A}$ are A-indexed sets. Is our current notion of morphism between instances (i.e. natural transformations) well-aligned with the above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

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

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

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

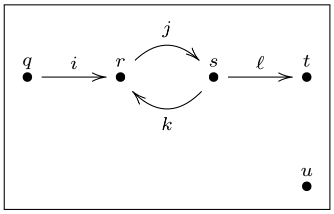
A database instance $I\colon \mathcal{G} \to \mathbf{Set}$ on $\mathcal{G}$ consists of two tables. Here is an example instance:

$I :=$ 

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

Vertex (I)
ID
v
w
x

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

$J :=$ 

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

Vertex (J)
ID
q
r
s
t
u

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

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

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

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

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

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

Vertex (Y_A)
ID
v_0
v_1

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

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

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

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

◊

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

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

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

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

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

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

4.3.4 Equivalence of categories

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

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

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

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

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

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

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

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

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

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

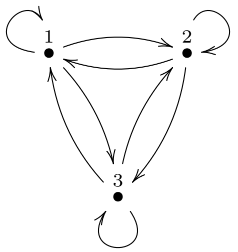
Exercise 4.3.4.2. Let $\mathcal{C}$ and $\mathcal{C}'$ be categories. Suppose that $F\colon \mathcal{C}\to\mathcal{C}'$ is an isomorphism of categories.

a.) Is it an equivalence of categories?

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

◊

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



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

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

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

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

for every set $S \neq \emptyset$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$s \xrightarrow{i_c} c \xrightarrow{g} c' \xrightarrow{j_{c'}} t.$$

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

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

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

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

□

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

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

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

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

◊

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

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

a.) Is it full?

b.) Is it faithful?

◊

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

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

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

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

◊

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

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

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

is bijective. Consider the following diagram

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

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

□

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

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

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

4.4.1 The category Sch of schemas

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

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

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

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

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

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

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

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

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

$\lozenge$

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

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

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

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

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

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

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

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

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

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

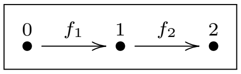
We now define the *category of schemas*, denoted **Sch**, to be the category whose objects are schemas as in Definition 3.5.2.6 and whose morphisms are schema morphisms defined as above. The identity morphism on schema $\mathcal{C} = (G, \simeq_G)$ is the schema morphism $\mathrm{id}_{\mathcal{C}} :=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

Slogan 4.4.1.3.

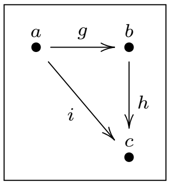
> " *A schema morphism sends vertices to vertices, arrows to paths, and path equivalences to path equivalences.* "

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

$[2] :=$



$\mathcal{C} :=$



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

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

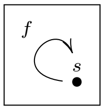
a.) How many schema morphisms are there $[2] \to \mathcal{C}$ that send 0 to a?

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

◊

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

$\mathcal{L}oop :=$



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

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

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

◊

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

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

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

◊

4.4.2 Proving the equivalence

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

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

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

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

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

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

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

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

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

Theorem 4.4.2.3. *The functors*

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

are mutually inverse equivalences of categories.

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

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

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

□

4.5 Limits and colimits

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

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

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

4.5.1 Products and coproducts in a category

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

4.5.1.1 Products

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

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

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

Note that the projection functions $X_1 \times X_2 \to X_1$ and $X_1 \times X_2 \to X_2$ induce morphisms of preorders. That is, if $(x_1, x_2) \leqslant_{1,2} (x'_1, x'_2)$ then in particular $x_1 \leqslant x'_1$. So we have preorder morphisms

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

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

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

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

$\Diamond$

Example 4.5.1.5. Given two graphs $G_1 = (V_1, A_1, src_1, tgt_1)$ and $G_2 = (V_2, A_2, src_2, tgt_2)$, we can take their product and get a new graph $G_1 \times G_2$. The vertices will be the grid of vertices $V_1 \times V_2$, so each vertex in $G_1 \times G_2$ is labeled by a pair of vertices, one from G_1 and one from G_2. When should an arrow connect (v_1, v_2) to (v'_1, v'_2)? Whenever we can find an arrow in G_1 connecting v_1 to v'_1 and we can find an arrow in G_2 connecting v_2 to v'_2. It turns out there is a simple formula for the set of arrows in $G_1 \times G_2$, namely $A_1 \times A_2$.

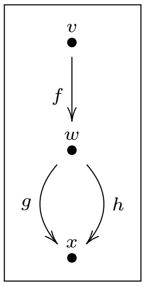
Let's write $G := G_1 \times G_2$ and say $G = (V, A, src, tgt)$. We now know that $V = V_1 \times V_2$ and $A = A_1 \times A_2$. What should the source and target functions $A \to V$ be? Given a function $src_1 \colon A_1 \to V_1$ and a function $src_2 \colon A_2 \to V_2$, the universal property of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

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

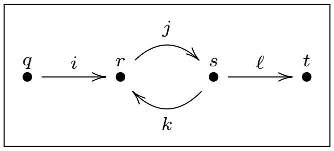
Namely the source of arrow (a_1, a_2) will be the vertex $(src_1(a_1), src_2(a_2))$. Similarly we have a ready-made choice of target function $tgt = tgt_1 \times tgt_2$. We have now defined the product graph.

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

$I :=$



$J :=$



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

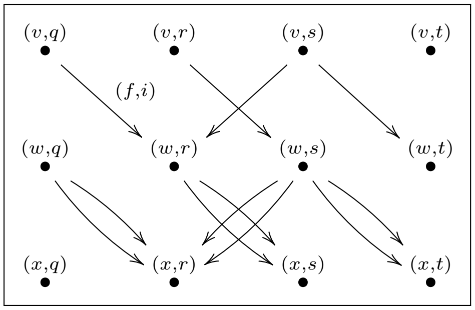
Vertex (I)
ID
v
w
x

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

Vertex (J)
ID
q
r
s
t

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

$I \times J :=$



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

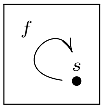
Vertex $(I \times J)$
ID
(v, q)
(v, r)
(v, s)
(v, t)
(w, q)
(w, r)
(w, s)
(w, t)
(x, q)
(x, r)
(x, s)
(x, t)

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

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

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

$\mathcal{L}oop :=$



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

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

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

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

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

◊

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

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

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

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

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

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

[23] The result is not necessarily inspiring, but at least computing it is straightforward.

[24] The names $X \times Y$ and π_1, π_2 are not mathematically important, they are pedagogically suggestive.

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

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

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

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

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

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

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

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

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

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

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

◊

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

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

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

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

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

◊

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

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

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

◊

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

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

The composition formula is "obvious".

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

$$[1] := \boxed{\overset{0}{\bullet} \xrightarrow{f} \overset{1}{\bullet}}$$

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

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

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

4.5.1.18 Coproducts

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

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

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

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

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

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

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

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

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

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

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

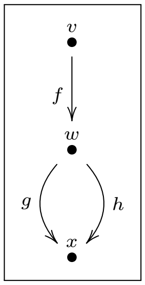
Let's write $G := G_1 \sqcup G_2$ and say $G = (V, A, src, tgt)$. We now know that $V = V_1 \sqcup V_2$ and $A = A_1 \sqcup A_2$. What should the source and target functions $A \to V$ be? Given a function $src_1 \colon A_1 \to V_1$ and a function $src_2 \colon A_2 \to V_2$, the universal property of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

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

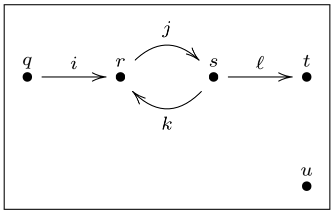
Namely for any arrow $a \in A$, we know either $a \in A_1$ or $a \in A_2$ (and not both), so the source of a will be the vertex $src_1(a)$ if $a \in A_1$ and $src_2(a)$ if $a \in A_2$. Similarly we have a ready-made choice of target function $tgt = tgt_1 \sqcup tgt_2$. We have now defined the coproduct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

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

$I :=$



$J :=$



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

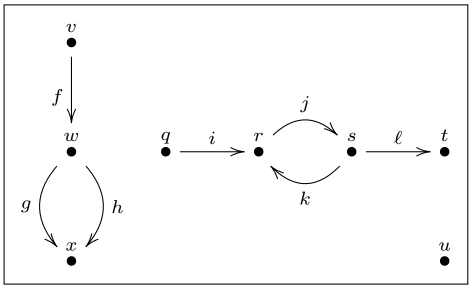
Vertex (I)
ID
v
w
x

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

Vertex (J)
ID
q
r
s
t
u

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

$I \sqcup J :=$



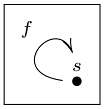
Arrow $(I \sqcup J)$		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x
i	q	r
j	r	s
k	s	r
ℓ	s	t

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

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

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

$\mathcal{L}oop :=$



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

a.) $\ulcorner$a diamond$\urcorner \sqcup \ulcorner$a heart$\urcorner$? $\ulcorner$a queen$\urcorner \sqcup \ulcorner$a black card$\urcorner$?

$\ulcorner$a card$\urcorner \sqcup \ulcorner$a red card$\urcorner$? $\ulcorner$a face card$\urcorner \sqcup \ulcorner$a black card$\urcorner$?

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

◊

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

Exercise 4.5.1.29. Consider the preorder $(\mathbb{N}, \texttt{divides})$, discussed in Exercise 4.5.1.4, where e.g. $5 \leqslant 15$ but $5 \nleqslant 6$.

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

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

◊

4.5.2 Diagrams in a category

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

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

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

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

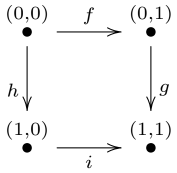
Example 4.5.2.2. Consider the commutative diagram in **Set** drawn below:

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

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

The fact that d is a functor means it must respect composition formulas, which implies that Diagram (4.18) commutes. Recall from Section 2.2 that not all diagrams one can draw will commute; one must specify that a given diagram commutes if he or she wishes to communicate this fact. But then how is a *non-commuting diagram* to be understood as a functor?

Let $G \in \mathrm{Ob}(\mathbf{Grph})$ denote the following graph



Recall the free category functor $F\colon \mathbf{Grph} \to \mathbf{Cat}$ from Example 4.1.2.30. The free category $F(G) \in \mathrm{Ob}(\mathbf{Cat})$ on G looks almost like $[1] \times [1]$ except that since $[f,g]$ is a different path in G than is $[h,i]$, they become different morphisms in $F(G)$. A functor $F(G) \to \mathbf{Set}$ might be drawn the same way that (4.18) is, but it would be a diagram that would *not* be said to commute.

We call $[1] \times [1]$ the *commutative square indexing category.* [28]

Exercise 4.5.2.3. Consider [2], the linear order category of length 2.

a.) Is [2] the appropriate indexing category for commutative triangles?

b.) If not, what is?

◊

[28] We might call what is here denoted by $F(G)$ the *noncommutative square indexing category.*

Example 4.5.2.4. Recall that an equalizer in **Set** was a diagram of sets that looked like this:

$$\begin{tikzcd}
\overset{E}{\bullet} \arrow[r, "f"] & \overset{A}{\bullet} \arrow[r, shift left, "g_1"] \arrow[r, shift right, "g_2"'] & \overset{B}{\bullet}
\end{tikzcd} \tag{4.19}$$

where $g_1 \circ f = g_2 \circ f$. What is the indexing category for such a diagram? It is the schema (4.19) with the PED $[f, g_1] \simeq [f, g_2]$. That is, in some sense you're seeing the indexing category, but the PED needs to be declared.

Exercise 4.5.2.5. Let $\mathcal{C}$ be a category, $A \in \mathrm{Ob}(\mathcal{C})$ an object, and $f\colon A \to A$ a morphism in $\mathcal{C}$. Consider the two diagrams in $\mathcal{C}$ drawn below:

$$\begin{tikzcd}
\overset{A}{\bullet} \arrow[r, "f"] & \overset{A}{\bullet} \arrow[r, "f"] & \overset{A}{\bullet} \arrow[r, "f"] & \cdots
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\overset{A}{\bullet} \arrow[loop left, "f"]
\end{tikzcd}$$

a.) Should these two diagrams have the same indexing category?

b.) If they should have the same indexing category, what is causing or allowing the pictures to appear different?

c.) If they should not have the same indexing category, what coincidence makes the two pictures have so much in common?

◊

Definition 4.5.2.6. Let $I \in \mathrm{Ob}(\mathbf{Cat})$ be a category. The *left cone on* I, denoted $I^{\triangleleft}$, is the category defined as follows. On objects we put $\mathrm{Ob}(I^{\triangleleft}) = \{-\infty\} \sqcup \mathrm{Ob}(I)$, and we call the new object $-\infty$ the *cone point of* $I^{\triangleleft}$. On morphisms we add a single new morphism $s_b\colon -\infty \to b$ for every object $b \in \mathrm{Ob}(I)$; more precisely,

$$\mathrm{Hom}_{I^{\triangleleft}}(a, b) = \begin{cases} \mathrm{Hom}_I(a, b) & \text{if } a, b \in \mathrm{Ob}(I) \\ \{s_b\} & \text{if } a = -\infty, b \in \mathrm{Ob}(I) \\ \{\mathrm{id}_{-\infty}\} & \text{if } a = b = -\infty \\ \varnothing & \text{if } a \in \mathrm{Ob}(I), b = -\infty. \end{cases}$$

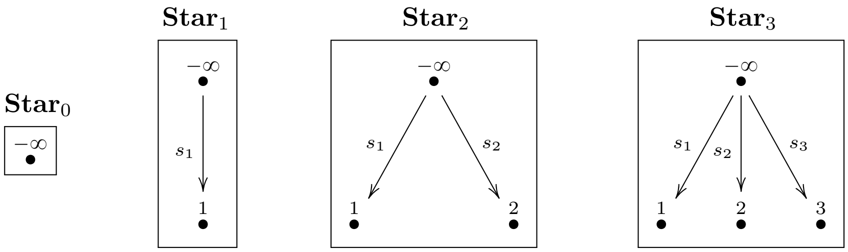
The composition formula is in some sense obvious. To compose two morphisms both in I, compose as dictated by I; if one has $-\infty$ as source then there will be a unique choice of composite.

There is an obvious inclusion of categories,

$$I \to I^{\triangleleft}. \tag{4.20}$$

Remark 4.5.2.7. Note that the specification of $I^{\triangleleft}$ given in Definition 4.5.2.6 works just as well if I is considered a schema and we are constructing a schema $I^{\triangleleft}$: add the new object $-\infty$ and the new arrows $s_b\colon -\infty \to b$ for each $b \in \mathrm{Ob}(I)$, and for every morphism $f\colon b \to b'$ in I add a PED $[s_{b'}] \simeq [s_b, f]$. We generally will not distinguish between categories and schemas, since they are equivalent.

Example 4.5.2.8. For a natural number $n \in \mathbb{N}$, we define the *n-leaf star schema*, denoted $\mathbf{Star}_n$, to be the category (or schema, see Remark 4.5.2.7) $\underline{n}^{\triangleleft}$, where $\underline{n}$ is the discrete category on n objects. Below we draw $\mathbf{Star}_0, \mathbf{Star}_1, \mathbf{Star}_2$, and $\mathbf{Star}_3$.



Exercise 4.5.2.9. Let $\mathcal{C}_0 := \underline{0}$ denote the empty category and for any natural number $n \in \mathbb{N}$, let $\mathcal{C}_{n+1} = (\mathcal{C}_n)^{\triangleleft}$. Draw $\mathcal{C}_4$. ◊

Exercise 4.5.2.10. Let $\mathcal{C}$ be the graph indexing schema as in (4.7). What is $\mathcal{C}^{\triangleleft}$ and how does it compare to (4.19)? ◊

Definition 4.5.2.11. Let $I \in \mathrm{Ob}(\mathbf{Cat})$ be a category. The *right cone on* I, denoted $I^{\triangleright}$, is the category defined as follows. On objects we put $\mathrm{Ob}(I^{\triangleright}) = \mathrm{Ob}(I) \sqcup \{\infty\}$, and we call the new object ∞ the *cone point of* $I^{\triangleright}$. On morphisms we add a single new morphism $t_b \colon b \to \infty$ for every object $b \in \mathrm{Ob}(I)$; more precisely,

$$\mathrm{Hom}_{I^{\triangleright}}(a,b) = \begin{cases} \mathrm{Hom}_I(a,b) & \text{if } a,b \in \mathrm{Ob}(I) \\ \{t_b\} & \text{if } a \in \mathrm{Ob}(I), b = \infty \\ \{\mathrm{id}_\infty\} & \text{if } a = b = \infty \\ \varnothing & \text{if } a = \infty, b \in \mathrm{Ob}(I). \end{cases}$$

The composition formula is in some sense obvious. To compose two morphisms both in I, compose as dictated by I; if one has ∞ as target then there will be a unique choice of composite.

There is an obvious inclusion of categories $I \to I^{\triangleright}$.

Exercise 4.5.2.12. Let $\mathcal{C}$ be the category $(\underline{2}^{\triangleleft})^{\triangleright}$, where $\underline{2}$ is the discrete category on two objects. Then $\mathcal{C}$ is somehow square-shaped, but what category is it exactly? Looking at Example 4.5.2.2, is $\mathcal{C}$ the commutative diagram indexing category $[1] \times [1]$, is it the non-commutative diagram indexing category $F(G)$, or is it something else? ◊

4.5.3 Limits and colimits in a category

Let $\mathcal{C}$ be a category, let I be an indexing category (which just means that I is a category that we're about to use as the indexing category for a diagram), and let $D \colon I \to \mathcal{C}$ an I-shaped diagram (which just means a functor). It is in relation to this setup that we can discuss the limit or colimit. In general the limit of a diagram $D \colon I \to \mathcal{C}$ will be a $I^{\triangleleft}$ shaped diagram $\lim D \colon I^{\triangleleft} \to \mathcal{C}$. In the case of products $I = \underline{2}$ and $I^{\triangleleft} = \mathbf{Star}_2$ looks like a span (see Example 4.5.2.8). But out of all the $I^{\triangleleft}$-shaped diagrams, which is the limit of D? Answer: the one with the universal "gateway" property, see Remark 4.5.1.9.

4.5.3.1 Universal objects

Definition 4.5.3.2. Let $\mathcal{C}$ be a category. An object $a \in \mathrm{Ob}(\mathcal{C})$ is called *initial* if, for all objects $c \in \mathrm{Ob}(\mathcal{C})$ there exists a unique morphism $a \to c$, i.e. $|\mathrm{Hom}_{\mathcal{C}}(a,c)| = 1$. An object $z \in \mathrm{Ob}(\mathcal{C})$ is called *terminal* if, for all objects $c \in \mathrm{Ob}(\mathcal{C})$ there is exists a unique morphism $c \to z$, i.e. $|\mathrm{Hom}_{\mathcal{C}}(c,z)| = 1$.

An object in a category is called *universal* if it is either initial or terminal, but we rarely use that term in practice, preferring to be specific about whether the object is initial or terminal. The word *final* is synonymous with the word terminal, but we'll try to constantly use terminal.

Colimits will end up being defined as initial things of a certain sort, and limits will end up being defined as terminal things of a certain sort. But we will get to that in Section 4.5.3.15.

Warning 4.5.3.3. A category $\mathcal{C}$ may have more than one initial object; similarly a category $\mathcal{C}$ may have more than one terminal object. We will see in Example 4.5.3.5 that any set with one element, e.g. $\{*\}$ or $\{☺\}$, is a terminal object in **Set**. These terminal sets have the same number of elements, but they are not the exact-same set; two sets having the same cardinality means precisely that there exists an isomorphism between them.

In fact, Proposition 4.5.3.4 below shows that in any category $\mathcal{C}$, any two terminal objects in $\mathcal{C}$ are isomorphic (similarly, any two initial objects in $\mathcal{C}$ are isomorphic). While there are many isomorphisms in **Set** between $\{1, 2, 3\}$ and $\{a, b, c\}$, there is only one isomorphism between $\{*\}$ and ☺. This is always the case for universal objects: there is a unique isomorphism between any two terminal (respectively initial) objects in any category.

As a result, people often speak of *the* initial object in $\mathcal{C}$ or *the* terminal object in $\mathcal{C}$, as though there was only one. "It's unique up to unique ismorphism!" is the justification for this use of the so-called definite article *the* rather than the indefinite article *a*. This is not a very misleading way of speaking, because just like the president today does not contain exactly the same atoms as the president yesterday, the difference is unimportant. But we still mention this as a warning: if $\mathcal{C}$ has a terminal object, we may speak of it as though it were unique, calling it *the terminal object*, and similarly for initial objects.

We will use the definite article throughout this document, e.g. in Example 4.5.3.5 we will discuss the initial object in **Set** and the terminal object in **Set**. This is common throughout mathematical literature as well.

Proposition 4.5.3.4. *Let $\mathcal{C}$ be a category and let $a_1, a_2 \in \mathrm{Ob}(\mathcal{C})$ both be initial objects. Then there is a unique isomorphism $a_1 \xrightarrow{\cong} a_2$. (Similarly, for any two terminal objects in $\mathcal{C}$ there is a unique isomorphism between them.)*

Proof. Suppose a_1 and a_2 are initial. Since a_1 is initial there is a unique morphism $f\colon a_1 \to a_2$; there is also a unique morphism $a_1 \to a_1$, which must be id_{a_1}. Since a_2 is initial there is a unique morphism $g\colon a_2 \to a_1$; there is also a unique morphism $a_2 \to a_2$, which must be id_{a_2}. So $g \circ f = \mathrm{id}_{a_1}$ and $f \circ g = \mathrm{id}_{a_2}$, which means that f is the desired (unique) isomorphism.

The proof for terminal objects is appropriately "dual".

□

Example 4.5.3.5. The initial object in **Set** is the set a for which there is always one way to map from a to anything else. Given $c \in \mathrm{Ob}(\textbf{Set})$ there is exactly one function $\varnothing \to c$, because there are no choices to be made, so the empty set $\varnothing$ is the initial object in **Set**.

The terminal object in **Set** is the set z for which there is always one way to map to z from anything else. Given $c \in \mathrm{Ob}(\textbf{Set})$ there is exactly one function $c \to \{☺\}$, where $\{☺\}$ is any set with one element, because there are no choices to be made: everything in c must be sent to the single element in $\{☺\}$. There are lots of terminal objects in **Set**, and they are all isomorphic to $\underline{1}$.

Example 4.5.3.6. The initial object in **Grph** is the graph a for which there is always one way to map from a to anything else. Given $c \in \mathrm{Ob}(\mathbf{Grph})$, there is exactly one function $\varnothing \to c$, where $\varnothing \in \mathbf{Grph}$ is the empty graph; so $\varnothing$ is the initial object.

The terminal object in **Grph** is more interesting. It is $\mathcal{L}oop$, the graph with one vertex and one arrow. In fact there are infinitely many terminal objects in **Grph**, but all of them are isomorphic to $\mathcal{L}oop$.

Exercise 4.5.3.7. Let X be a set, let $\mathbb{P}(X)$ be the set of subsets of X (see Definition 2.7.4.9). We can regard $\mathbb{P}(X)$ as a preorder under inclusion of subsets (see for example Section 3.4.2). And we can regard preorders as categories using a functor $\mathbf{PrO} \to \mathbf{Cat}$ (see Proposition 4.2.1.17).

a.) What is the initial object in $\mathbb{P}(X)$?

b.) What is the terminal object in $\mathbb{P}(X)$?

◊

Example 4.5.3.8. The initial object in the category **Mon** of monoids is the trivial monoid, $\underline{1}$. For any monoid M, a morphism of monoids $\underline{1} \to M$ is a functor between 1-object categories and these are determined by where they send morphisms. Since $\underline{1}$ has only the identity morphism and functors must preserve identities, there is no choice involved in finding a monoid morphism $\underline{1} \to M$.

Similarly, the terminal object in **Mon** is also the trivial monoid, $\underline{1}$. For any monoid M, a morphism of monoids $M \to \underline{1}$ sends everything to the identity; there is no choice.

Exercise 4.5.3.9.

a.) What is the initial object in **Grp**, the category of groups?

b.) What is the terminal object in **Grp**?

◊

Example 4.5.3.10. Recall the preorder **Prop** of logical propositions from Section 4.2.4.1. The initial object is a proposition that implies all others. It turns out that "FALSE" is such a proposition. The proposition "FALSE" is like "$1 \neq 1$"; in logical formalism it can be shown that if "FALSE" is true then everything is true.

The terminal object in **Prop** is a proposition that is implied by all others. It turns out that "TRUE" is such a proposition. In logical formalism, everything implies that "TRUE" is true.

Example 4.5.3.11. The discrete category $\underline{2}$ has no initial object and no terminal object. The reason is that it has two objects $1, 2$, but no maps from one to the other, so $\mathrm{Hom}_{\underline{2}}(1,2) = \mathrm{Hom}_{\underline{2}}(2,1) = \varnothing$.

Exercise 4.5.3.12. Recall the `divides` preorder from Exercise 4.5.1.4, where 5 `divides` 15.

a.) Considering this preorder as a category, does it have an initial object?

b.) Does it have a terminal object?

◊

Exercise 4.5.3.13. Let $\mathcal{M} = (\mathrm{List}(\{a,b\}), [\ ], +\!\!+)$ denote the free monoid on $\{a,b\}$ (see Definition 3.1.1.15), considered as a category (via Theorem 4.2.1.3).

a.) Does it have an initial object?

b.) Does it have a terminal object?

c.) Which monoids have initial (respectively terminal) objects?

$\Diamond$

Exercise 4.5.3.14. Let S be a set and consider the indiscrete category $K_S \in \mathrm{Ob}(\mathbf{Cat})$ on objects S (see Example 4.3.4.3).

a.) For what S does K_S have an initial object?

b.) For what S does K_S have a terminal object?

$\Diamond$

4.5.3.15 Examples of limits

Let $\mathcal{C}$ be a category and let $X, Y \in \mathrm{Ob}(\mathcal{C})$ be objects. Definition 4.5.1.8 defines a product of X and Y to be a span $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$ such that for every other span $X \xleftarrow{p} Z \xrightarrow{q} Y$ there exists a unique morphism $Z \to X \times Y$ making the triangles commute. It turns out that we can enunciate this in our newly formed language of universal objects by saying that the span $X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y$ is itself a terminal object in the category of spans on X and Y. Phrasing the definition of products in this way will be generalizable to defining arbitrary limits.

Construction 4.5.3.16 (Products). Let $\mathcal{C}$ be a category and let X_1, X_2 be objects. We can consider this setup as a diagram $X \colon \underline{2} \to \mathcal{C}$, where $X(1) = X_1$ and $X(2) = X_2$. Consider the category $\underline{2}^{\lhd} = \mathbf{Star}_2$, which is drawn in Example 4.5.2.8; the inclusion $i \colon \underline{2} \to \underline{2}^{\lhd}$, as in (4.20); and the category of functors $\mathrm{Fun}(\underline{2}^{\lhd}, \mathcal{C})$. The objects in $\mathrm{Fun}(\underline{2}^{\lhd}, \mathcal{C})$ are spans in $\mathcal{C}$ and the morphisms are natural transformations between them. Given a functor $S \colon \underline{2}^{\lhd} \to \mathcal{C}$ we can compose with $i \colon \underline{2} \to \underline{2}^{\lhd}$ to get a functor $\underline{2} \to \mathcal{C}$. We want that to be X.

$$\begin{tikzcd}
\underline{2} \arrow[r, "X"] \arrow[d, "i"'] & \mathcal{C} \\
\underline{2}^{\lhd} \arrow[ur, "S"'] &
\end{tikzcd}$$

So we are ready to define the category of spans on X_1 and X_2.

Define the *category of spans on* X, denoted $\mathcal{C}_{/X}$, to be the category whose objects and morphisms are as follows:

$$\begin{aligned}
\mathrm{Ob}(\mathcal{C}_{/X}) &= \{S \colon \underline{2}^{\lhd} \to \mathcal{C} \mid S \circ i = X\} \\
\mathrm{Hom}_{\mathcal{C}_{/X}}(S, S') &= \{\alpha \colon S \to S' \mid \alpha \circ i = \mathrm{id}_X\}.
\end{aligned} \tag{4.21}$$

The product of X_1 and X_2 was defined in Definition 4.5.1.8; we can now recast $X_1 \times X_2$ as the terminal object in $\mathcal{C}_{/X}$.

To bring this down to earth, an object in $\mathcal{C}_{/X}$ can be pictured as a diagram in $\mathcal{C}$ of the following form:

$$\begin{tikzcd}
& Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\
X_1 & & X_2
\end{tikzcd}$$

In other words, the objects of $\mathcal{C}_{/X}$ are spans, each of which we might write in-line as $X_1 \xleftarrow{p} Z \xrightarrow{q} X_2$. A morphism in $\mathcal{C}_{/X}$ from object $X_1 \xleftarrow{p} Z \xrightarrow{q} X_2$ to object $X_1 \xleftarrow{p'} Z' \xrightarrow{q'} X_2$ consists of a morphism $\ell \colon Z \to Z'$, such that $p' \circ \ell = p$ and $q' \circ \ell = q$. So the set of such morphisms in $\mathcal{C}_{/X}$ are all the ℓ's that make the right-hand diagram commute: [29]

$$\mathrm{Hom}_{\mathcal{C}_{/X}}\left(\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\ X_1 & & X_2 \end{tikzcd}, \quad \begin{tikzcd} & Z' \arrow[dl, "p'"'] \arrow[dr, "q'"] & \\ X_1 & & X_2 \end{tikzcd}\right) = \left\{\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] \arrow[dd, dashed, "\ell"] & \\ X_1 & & X_2 \\ & Z' \arrow[ul, "p'"] \arrow[ur, "q'"'] & \end{tikzcd}\right\} \tag{4.22}$$

Each object in $\mathcal{C}_{/X}$ is a span on X_1 and X_2, and each morphism in $\mathcal{C}_{/X}$ is a "morphism of cone points in $\mathcal{C}$ making everything in sight commute". The terminal object in $\mathcal{C}_{/X}$ is the product of X_1 and X_2; see Definition 4.5.1.8.

It may be strange to have a category in which the objects are spans in another category. But once you admit this possibility, the notion of morphism between spans is totally sensible. Or if it isn't, then stare at (4.22) for 30 seconds and say to yourself "When in Rome..!" These are the aqueducts of category theory, and they work wonders.

Example 4.5.3.17. Consider the arbitrary 6-object category $\mathcal{C}$ drawn below, in which the

[29] To be completely pedantic, according to (4.21), the morphisms in $\mathcal{C}_{/X}$ should be drawn like this:

$$\mathrm{Hom}_{\mathcal{C}_{/X}}\left(\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] & \\ X_1 & & X_2 \end{tikzcd}, \quad \begin{tikzcd} & Z' \arrow[dl, "p'"'] \arrow[dr, "q'"] & \\ X_1 & & X_2 \end{tikzcd}\right) = \left\{\begin{tikzcd} & Z \arrow[dl, "p"'] \arrow[dr, "q"] \arrow[ddd, dashed, "\alpha_{-\infty}"] & \\ X_1 \arrow[d, equal, "\alpha_1"'] & & X_2 \arrow[d, equal, "\alpha_2"] \\ X_1 & & X_2 \\ & Z' \arrow[ul, "p'"] \arrow[ur, "q'"'] & \end{tikzcd}\right\}$$

But this is going a bit overboard. The point is, the set $\mathrm{Hom}_{\mathcal{C}_{/X}}$ is the set of morphisms serving the role of $\alpha_{-\infty} \colon Z \to Z'$.

three diagrams that can commute do:

$$\begin{tikzcd}
 & & \checkmark & X_1 & \checkmark & \\
\mathcal{C} := & A \arrow[urr, bend left, "a"] & B \arrow[l, "f"'] \arrow[ur, "b_1"] \arrow[dr, "b_2"'] & & C \arrow[ul, "c_1"'] \arrow[dl, "c_2"] \arrow[r, "g"] & D \arrow[ull, bend right, "d_1"'] \arrow[dll, bend left, "d_2"] \\
 & & & X_2 & \checkmark &
\end{tikzcd}$$

Let $X\colon \underline{2} \to \mathcal{C}$ be given by $X(1) = X_1$ and $X(2) = X_2$. Then the category of spans on X might be drawn

$$\mathcal{C}_{/X} \cong \begin{tikzcd}
(B,b_1,b_2) & (C,c_1,c_2) \arrow[r, "g"] & (D,d_1,d_2)
\end{tikzcd}$$

4.5.3.18 Definition of limit

Definition 4.5.3.19. Let $\mathcal{C}$ be a category, let I be a category; let $I^{\lhd}$ be the left cone on I, and let $i\colon I \to I^{\lhd}$ be the inclusion. Suppose that $X\colon I \to \mathcal{C}$ is an I-shaped diagram in $\mathcal{C}$. The *slice category of $\mathcal{C}$ over X* denoted $\mathcal{C}_{/X}$ is the category whose objects and morphisms are as follows:

$$\mathrm{Ob}(\mathcal{C}_{/X}) = \{S\colon I^{\lhd} \to \mathcal{C} \mid S \circ i = X\}$$
$$\mathrm{Hom}_{\mathcal{C}_{/X}}(S, S') = \{\alpha\colon S \to S' \mid \alpha \circ i = \mathrm{id}_X\}.$$

A *limit of X*, denoted $\lim_I X$ or $\lim X$, is a terminal object in $\mathcal{C}_{/X}$.

Pullbacks The relevant indexing category for pullbacks is the cospan, $I = \underline{2}^{\rhd}$ drawn as to the left below:

$$I \qquad \begin{tikzcd}
0 \arrow[dr] & & 1 \arrow[dl] \\
 & 2 &
\end{tikzcd}
\qquad\qquad
X\colon I \to \mathcal{C} \qquad \begin{tikzcd}
X_0 \arrow[dr] & & X_1 \arrow[dl] \\
 & X_2 &
\end{tikzcd}$$ [30]

A I-shaped diagram in $\mathcal{C}$ is a functor $X\colon I \to \mathcal{C}$, which we might draw as to the right above (e.g. $X_0 \in \mathrm{Ob}(\mathcal{C})$).

[30] We use a dash box here because we're not drawing the whole category but merely a diagram existing inside $\mathcal{C}$.

An object S in the slice category $\mathcal{C}_{/X}$ is a commutative diagram $S\colon I^{\triangleleft}\to\mathcal{C}$ over X, which looks like the box to the left below:

$f\colon S\to S'$

$S\in\mathrm{Ob}(\mathcal{C}_{/X})$

$$\begin{tikzcd}
 & S_{-\infty} \arrow[dl] \arrow[dr] & \\
\overset{X_0}{\bullet} \arrow[dr] & & \overset{X_1}{\bullet} \arrow[dl] \\
 & \overset{X_2}{\bullet} &
\end{tikzcd}$$

$$\begin{tikzcd}
 & S_{-\infty} \arrow[d, "f"] \arrow[ddl, bend right] \arrow[ddr, bend left] & \\
 & S'_{-\infty} \arrow[dl] \arrow[dr] & \\
\overset{X_0}{\bullet} \arrow[dr] & & \overset{X_1}{\bullet} \arrow[dl] \\
 & \overset{X_2}{\bullet} &
\end{tikzcd}$$

A morphism in $\mathcal{C}_{/X}$ is drawn in the dashbox to the right above. A terminal object in $\mathcal{C}_{/X}$ is precisely the "gateway" we want, i.e. the limit of X is the pullback $X_0\times_{X_2}X_1$.

Exercise 4.5.3.20. Let I be the graph indexing category (see 4.7).

a.) What is $I^{\triangleleft}$?

b.) Now let $G\colon I\to\mathbf{Set}$ be the graph from Example 3.3.1.2. Give an example of an object in $\mathbf{Set}_{/G}$.

c.) We have already given a name to the limit of $G\colon I\to\mathbf{Set}$; what is it?

◊

Exercise 4.5.3.21. Let $\mathcal{C}$ be a category and let $I=\varnothing$ be the empty category. There is a unique functor $X\colon\varnothing\to\mathcal{C}$.

a.) What is the slice category $\mathcal{C}_{/X}$?

b.) What is the limit of X?

◊

Example 4.5.3.22. Often one wants to take the limit of some strange diagram. We have now constructed the limit for any shape diagram. For example, if we want to take the product of more than two, say n, objects, we could use the diagram shape $I=\underline{n}$ whose cone is $\mathbf{Star}_n$ from Example 4.5.2.8.

Example 4.5.3.23. We have now defined limits in any category, so we have defined limits in $\mathbf{Cat}$. Let $[1]$ denote the category depicted

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

and let $\mathcal{C}$ be a category. Naming two categories is the same thing as naming a functor $X\colon\underline{2}\to\mathbf{Cat}$, so we now have such a functor. Its limit is denoted $[1]\times\mathcal{C}$. It turns out that $[1]\times\mathcal{C}$ looks like a "$\mathcal{C}$-shaped prism". It consists of two panes, front and back say, each having the precise shape as $\mathcal{C}$ (same objects, same arrows, same composition), and morphisms from the front pane to the back pane making all front-to-back squares commute. For example, if $\mathcal{C}$ looked was the category generated by the schema to the left below, then $\mathcal{C} \times [1]$ would be the category generated by the schema to the right below:

$$\begin{tikzcd}
\overset{A}{\bullet} \arrow[r, "f"] \arrow[d, "g"'] & \overset{B}{\bullet} \arrow[d, "h"] \\
\overset{C}{\bullet} & \overset{D}{\bullet}
\end{tikzcd}
\qquad
\begin{tikzcd}
& \overset{A1}{\bullet} \arrow[rr, "f1"] \arrow[dd, "g1"'] & & \overset{B1}{\bullet} \arrow[dd, "h1"] \\
\overset{A0}{\bullet} \arrow[ur, "Ae"] \arrow[rr, "f0"] \arrow[dd, "g0"'] & & \overset{B0}{\bullet} \arrow[ur, "Be"] \arrow[dd, "h0"] & \\
& \overset{C1}{\bullet} & & \overset{D1}{\bullet} \\
\overset{C0}{\bullet} \arrow[ur, "Ce"'] & & \overset{D0}{\bullet} \arrow[ur, "De"'] &
\end{tikzcd}$$

It turns out that a natural transformation $\alpha \colon F \to G$ between functors $F, G \colon \mathcal{C} \to \mathcal{D}$ is the same thing as a functor $\mathcal{C} \times [1] \to \mathcal{D}$ such that the front pane is sent via F and the back pane is sent via G. The components are captured by the front-to-back morphisms, and the naturality is captured by the commutativity of the front-to-back squares in $\mathcal{C} \times [1]$.

Remark 4.5.3.24. Recall in Section 2.7.6.6 we described relative sets. In fact, Definition 2.7.6.7 basically defines a category of relative sets over any fixed set B. Let $\underline{1}$ denote the discrete category on one object, and note that providing a functor $\underline{1} \to \mathbf{Set}$ is the same as simply providing a set, so consider $B \colon \underline{1} \to \mathbf{Set}$. Then the slice category $\mathbf{Set}_{/B}$, as defined in Definition 4.5.3.19 is precisely the category of relative sets over B: it has the same objects and morphisms as was described in Definition 2.7.6.7.

4.5.3.25 Definition of colimit

The definition of colimits is appropriately "dual" to the definition of limits. Instead of looking at left cones, we look at right cones; instead of being interested in terminal objects, we are interested in initial objects.

Definition 4.5.3.26. Let $\mathcal{C}$ be a category, let I be a category; let $I^{\rhd}$ be the right cone on I, and let $i \colon I \to I^{\rhd}$ be the inclusion. Suppose that $X \colon I \to \mathcal{C}$ is an I-shaped diagram in $\mathcal{C}$. The *coslice category of* $\mathcal{C}$ *over* X denoted $\mathcal{C}_{X/}$ is the category whose objects and morphisms are as follows:

$$\mathrm{Ob}(\mathcal{C}_{X/}) = \{S \colon I^{\rhd} \to \mathcal{C} \mid S \circ i = X\}$$
$$\mathrm{Hom}_{\mathcal{C}_{X/}}(S, S') = \{\alpha \colon S \to S' \mid \alpha \circ i = \mathrm{id}_X\}.$$

A *colimit of* X, denoted $\mathrm{colim}_I\, X$ or $\mathrm{colim}\, X$, is an initial object in $\mathcal{C}_{X/}$.

Pushouts The relevant indexing category for pushouts is the span, $I = \underline{2}^{\triangleleft}$ drawn as to the left below:

I

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

$X\colon I \to \mathcal{C}$

$$\begin{tikzcd}
\overset{X_1}{\bullet} & & \overset{X_2}{\bullet} \\
& \overset{X_0}{\bullet} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

An I-shaped diagram in $\mathcal{C}$ is a functor $X\colon I \to \mathcal{C}$, which we might draw as to the right above (e.g. $X_0 \in \mathrm{Ob}(\mathcal{C})$).

An object S in the coslice category $\mathcal{C}_{X/}$ is a commutative diagram $S\colon I^{\triangleright} \to \mathcal{C}$ over X, which looks like the box to the left below:

$S \in \mathrm{Ob}(\mathcal{C}_{X/})$

$$\begin{tikzcd}
& S_\infty & \\
\overset{X_1}{\bullet} \arrow[ur] & & \overset{X_2}{\bullet} \arrow[ul] \\
& \overset{X_0}{\bullet} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

$f\colon S \to S'$

$$\begin{tikzcd}
& S'_\infty & \\
& S_\infty \arrow[u, "f"'] & \\
\overset{X_1}{\bullet} \arrow[ur] \arrow[uur, bend left] & & \overset{X_2}{\bullet} \arrow[ul] \arrow[uul, bend right] \\
& \overset{X_0}{\bullet} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

A morphism in $\mathcal{C}_{X/}$ is drawn in the dashbox to the right above. An initial object in $\mathcal{C}_{X/}$ is precisely the "gateway" we want; i.e. the colimit of X is the pushout, $X_1 \sqcup_{X_0} X_2$.

Exercise 4.5.3.27. Let I be the graph indexing category (see 4.7).

a.) What is $I^{\triangleright}$?

b.) Now let $G\colon I \to \mathbf{Set}$ be the graph from Example 3.3.1.2. Give an example of an object in $\mathbf{Set}_{G/}$.

c.) We have already given a name to the colimit of $G\colon I \to \mathbf{Set}$; what is it?

◊

Exercise 4.5.3.28. Let $\mathcal{C}$ be a category and let $I = \varnothing$ be the empty category. There is a unique functor $X\colon \varnothing \to \mathcal{C}$.

a.) What is the coslice category $\mathcal{C}_{X/}$?

b.) What is the colimit of X (assuming it exists)?

◊

Example 4.5.3.29 (Cone as colimit). We have now defined colimits in any category, so we have defined colimits in **Cat**. Let $\mathcal{C}$ be a category and recall from Example 4.5.3.23 the category $\mathcal{C} \times [1]$. The inclusion of the front pane is a functor $i_0 \colon \mathcal{C} \to \mathcal{C} \times [1]$ (similarly, the inclusion of the back pane is a functor $i_1 \colon \mathcal{C} \to \mathcal{C} \times [1]$). Finally let $t \colon \mathcal{C} \to \underline{1}$ be the unique functor to the terminal category (see Exercise 4.1.2.37). We now have a diagram in **Cat** of the form

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "i_0"] \arrow[d, "t"'] & \mathcal{C} \times [1] \\
\underline{1} &
\end{tikzcd}$$

The colimit (i.e. the pushout) of this diagram in **Cat** slurps down the entire front pane of $\mathcal{C} \times [1]$ to a point, and the resulting category is isomorphic to $\mathcal{C}^{\triangleleft}$. Figure 4.23 is a drawing of this phenomenon.

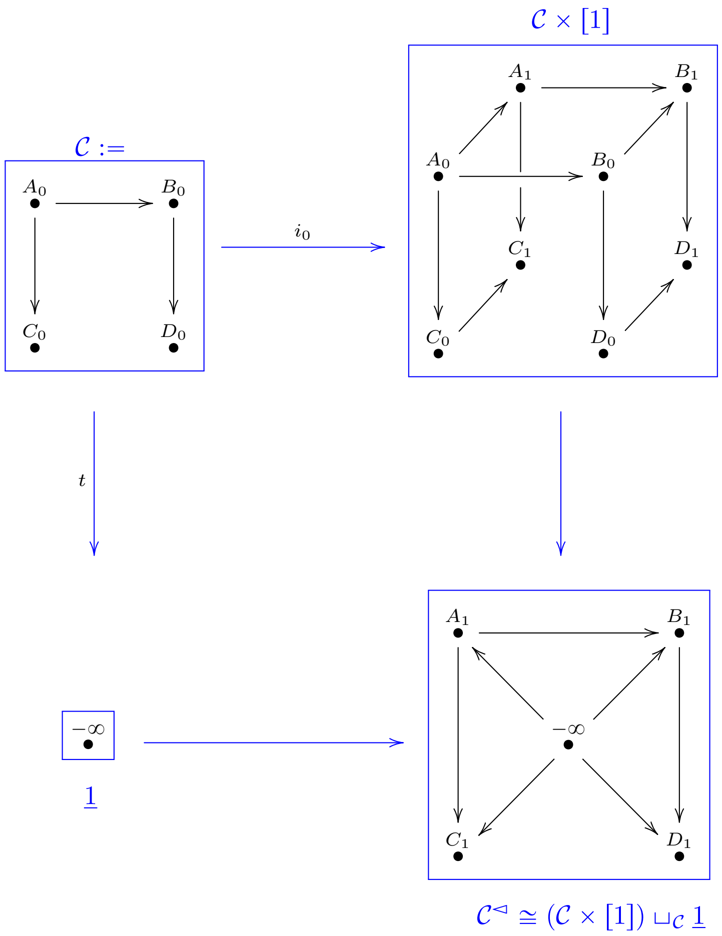
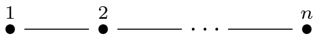


Figure 4.23: Let $\mathcal{C}$ be the category drawn in the upper left corner. The left cone $\mathcal{C}^{\triangleleft}$ on $\mathcal{C}$ is obtained as a pushout in **Cat**. We first make a prism $\mathcal{C} \times [1]$, and then identify the front pane with a point.

(Similarly, the pushout of the analogous diagram for i_1 would give $\mathcal{C}^{\triangleright}$.)

Example 4.5.3.30. Consider the category **Top** of topological spaces. The (hollow) circle is a topological space which people often denote S^1 (for "1-dimensional sphere"). The filled-in circle, also called a 2-dimensional disk, is denoted D^2. The inclusion of the circle into the disk is continuous so we have a morphism in **Top** of the form $i\colon S^1 \to D^2$. The terminal object in **Top** is the one-point space $\{\smiley\}$, and so there is a unique morphism $t\colon S^1 \to \{\smiley\}$. The pushout of the diagram $D^2 \xleftarrow{i} S^1 \xrightarrow{t} \{\smiley\}$ is isomorphic to the 2-dimensional sphere (the exterior of a tennis ball), S^2. The reason is that we have slurped the entire bounding circle to a point, and the category of topological spaces has the right morphisms to ensure that the resulting space really is a sphere.

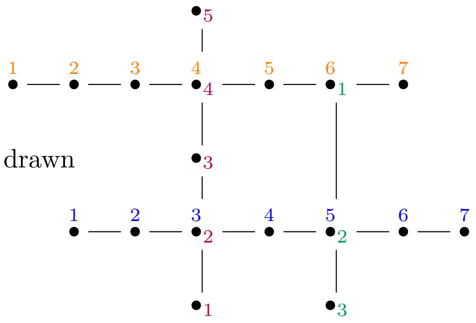
Application 4.5.3.31. Consider the symmetric graph G_n consisting of a chain of n vertices,



Think of this as modeling a subway line. There are n-many graph homomorphisms $G_1 \to G_n$ given by the various vertices. One can create transit maps using colimits. For example, the colimit of the diagram to the left is the symmetric graph drawn to the right below.

$$\mathrm{colim}\left(\begin{tikzcd} G_1 \arrow[r, "4"] \arrow[d, "4"'] & G_7 & G_1 \arrow[l, "6"'] \arrow[d, "1"] \\ G_5 & & G_3 \\ G_1 \arrow[u, "2"] \arrow[r, "3"'] & G_7 & G_1 \arrow[l, "5"] \arrow[u, "2"'] \end{tikzcd}\right)$$

can be drawn



◊◊

4.6 Other notions in Cat

In this section we discuss some leftover notions about categories. For example in Section 4.6.1 we explain a kind of duality for categories, in which arrows are flipped. For example reversing the order in a preorder is an example of this duality, as is the similarity between limits and colimits. In Section 4.6.2 we discuss the so-called Grothendieck construction which in some sense graphs functors, and we show that it is useful for transforming databases into the kind of format (RDF) used in scraping data off webpages. We define a general construction for creating categories in Section 4.6.4. Finally, in Section 4.6.5 we show that precisely the same arithmetic statements that held for sets in Section 2.7.3 hold for categories.

4.6.1 Opposite categories

People used to discuss two different kinds of functors between categories: the so-called *covariant functors* and the so-called *contravariant functors*. Covariant functors are what we have been calling functors. The reader may have come across the idea of contravariance when considering Exercise 4.2.3.2.[31] There we saw that a continuous mapping of topological spaces $f\colon X \to Y$ does not induce a morphism of orders on their open sets $\mathrm{Open}(X) \to \mathrm{Open}(Y)$; that is not required by the notion of continuity. Instead, a morphism of topological spaces $f\colon X \to Y$ induces a morphism of orders $\mathrm{Open}(Y) \to \mathrm{Open}(X)$, going backwards. So we do not have a functor **Top** → **PrO** in this way, but it's quite close. One used to say that Open is a *contravariant functor* **Top** → **PrO**.

[31] Similarly, see Exercise 4.2.4.4.

As important and common as contravariance is, people found that keeping track of which functors were covariant and which were contravariant was a big hassle. Luckily, there is a simple work-around, which simplifies everything: the notion of opposite categories.

Definition 4.6.1.1. Let $\mathcal{C}$ be a category. The *opposite category* of $\mathcal{C}$, denoted $\mathcal{C}^{\rm op}$, has the same objects as $\mathcal{C}$, i.e. $\mathrm{Ob}(\mathcal{C}^{\rm op}) = \mathrm{Ob}(\mathcal{C})$, and for any two objects c, c', one defines

$$\mathrm{Hom}_{\mathcal{C}^{\rm op}}(c, c') := \mathrm{Hom}_{\mathcal{C}}(c', c).$$

Example 4.6.1.2. If $n \in \mathbb{N}$ is a natural number and $\underline{n}$ the corresponding discrete category, then $\underline{n}^{\rm op} = \underline{n}$. Recall the span category $I = \underline{2}^{\triangleleft}$ from Definition 4.5.1.8. Its opposite is the cospan category $I^{\rm op} = \underline{2}^{\triangleright}$, from Definition 4.5.1.23.

Exercise 4.6.1.3. Let $\mathcal{C}$ be the category from Example 4.5.3.17. Draw $\mathcal{C}^{\rm op}$. ◊

Lemma 4.6.1.4. *Let $\mathcal{C}$ and $\mathcal{D}$ be categories. One has $(\mathcal{C}^{\rm op})^{\rm op} = \mathcal{C}$. Also we have $\mathrm{Fun}(\mathcal{C}, \mathcal{D}) \cong \mathrm{Fun}(\mathcal{C}^{\rm op}, \mathcal{D}^{\rm op})$. This implies that a functor $\mathcal{C}^{\rm op} \to \mathcal{D}$ can be identified with a functor $\mathcal{C} \to \mathcal{D}^{\rm op}$.*

Proof. This follows straightforwardly from the definitions.

□

Exercise 4.6.1.5. In Exercises 4.2.3.2, 4.2.4.3, and 4.2.4.4 there were questions about whether a certain function $\mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{D})$ extended to a functor $\mathcal{C} \to \mathcal{D}$. In each case, see if the proposed function would extend to a "contravariant functor" i.e. to a functor $\mathcal{C}^{\rm op} \to \mathcal{D}$. ◊

Example 4.6.1.6 (Simplicial sets). Recall from Example 4.3.4.4 the category $\mathbf{\Delta}$ of linear orders $[n]$. For example, $[1]$ is the linear order $0 \leqslant 1$ and $[2]$ is the linear order $0 \leqslant 1 \leqslant 2$. Both $[1]$ and $[2]$ are objects of $\mathbf{\Delta}$. There are 6 morphisms from $[1]$ to $[2]$, which we could denote

$$\mathrm{Hom}_{\mathbf{\Delta}}([1], [2]) = \{(0,0), (0,1), (0,2), (1,1), (1,2), (2,2)\}.$$

It may seem strange, but the category $\mathbf{\Delta}^{\rm op}$ turns out to be quite useful in algebraic topology. It is the indexing category for a combinatorial approach to the homotopy theory of spaces. That is, we can represent something like the category of spaces and continuous maps using the functor category $\mathbf{sSet} := \mathrm{Fun}(\mathbf{\Delta}^{\rm op}, \mathbf{Set})$, which is called the *category of simplicial sets*.

This may seem very complicated compared to something we did earlier, namely simplicial complexes. But simplicial sets have excellent formal properties that simplicial complexes do not. We will not go further with this here, but through the work of Dan Kan, André Joyal, Jacob Lurie, and many others, simplicial sets have allowed category theory to pierce deeply into the realm of topology and vice versa.

4.6.2 Grothendieck construction

Let $\mathcal{C}$ be a database schema (or category) and let $J \colon \mathcal{C} \to \mathbf{Set}$ be an instance. We have been drawing this in table form, but there is another standard way of laying out the data in J, called the *resource descriptive framework* or RDF. Developed for the web, RDF is a useful format when one does not have a schema in hand, e.g. when scraping information off of a website, one does not know what schema will be best. In these cases, information is stored in so-called RDF triples, which are of the form

$$\langle \text{Subject, Predicate, Object} \rangle$$

For example, one might see something like

Subject	**Predicate**	**Object**
A01	occurredOn	D13114
A01	performedBy	P44
A01	actionDescription	Told congress to raise debt ceiling
D13114	hasYear	2013
D13114	hasMonth	January
D13114	hasDay	14
P44	FirstName	Barack
P44	LastName	Obama

(4.24)

Category-theoretically, it is quite simple to convert a database instance $J\colon \mathcal{C} \to \mathbf{Set}$ into an RDF triple store. To do so, we use the *Grothendieck construction*, which is more aptly named the category of elements construction, defined below.[32]

Definition 4.6.2.1. Let $\mathcal{C}$ be a category and let $J\colon \mathcal{C} \to \mathbf{Set}$ be a functor. The *category of elements of* J, denoted $\int_{\mathcal{C}} J$, is defined as follows:

$$\begin{aligned} \mathrm{Ob}(\int_{\mathcal{C}} J) &:= \{(C, x) \mid C \in \mathrm{Ob}(\mathcal{C}), x \in J(C)\}. \\ \mathrm{Hom}_{\int_{\mathcal{C}} J}((C, x), (C', x')) &:= \{f\colon C \to C' \mid J(f)(x) = x'\}. \end{aligned}$$

There is a natural functor $\pi_J\colon \int_{\mathcal{C}} J \longrightarrow \mathcal{C}$. It sends each object $(C, x) \in \mathrm{Ob}(\int_{\mathcal{C}} J)$ to the object $C \in \mathrm{Ob}(\mathcal{C})$. And it sends each morphism $f\colon (C, x) \to (C', x')$ to the morphism $f\colon C \to C'$. We call π_J the *projection functor*.

Example 4.6.2.2. Let A be a set, and consider it as a discrete category. We saw in Exercise 4.3.3.4 that a functor $S\colon A \to \mathbf{Set}$ is the same thing as an A-indexed set, as discussed in Section 2.7.6.10. We will follow Definition 2.7.6.12 and for each $a \in A$ write $S_a := S(a)$.

What is the category of elements of a functor $S\colon A \to \mathbf{Set}$? The objects of $\int_A S$ are pairs (a, s) where $a \in A$ and $s \in S(a)$. Since A has nothing but identity morphisms, $\int_A S$ has nothing but identity morphisms; i.e. it is the discrete category on a set. In fact that set is the disjoint union

$$\int_A S = \bigsqcup_{a \in A} S_a.$$

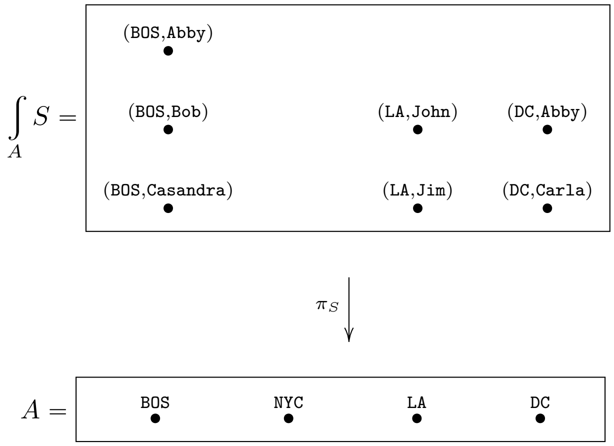
The functor $\pi_S\colon \int_A S \to A$ sends each element in S_a to the element $a \in A$.

[32] Apparently, Alexander Grothendieck did not invent this construction, it was discussed prior to Grothendieck's use of it, e.g. by Mac Lane. But more to the point, the term Grothendieck construction is not grammatically suited in the sense that both the following are awkward in English: "the Grothendieck construction of J is ..." (awkward because J is not being constructed but used in a construction) and "the Grothendieck construct for J is..." (awkward because it just is). The term *category of elements* is more descriptive and easier to use grammatically.

One can see this as a kind of histogram. For example, let $A = \{\texttt{BOS}, \texttt{NYC}, \texttt{LA}, \texttt{DC}\}$ and let $S\colon A \to \textbf{Set}$ assign

$$\begin{aligned} S_{\texttt{BOS}} &= \{\texttt{Abby}, \texttt{Bob}, \texttt{Casandra}\}, \\ S_{\texttt{NYC}} &= \varnothing, \\ S_{\texttt{LA}} &= \{\texttt{John}, \texttt{Jim}\}, \text{and} \\ S_{\texttt{DC}} &= \{\texttt{Abby}, \texttt{Carla}\}. \end{aligned}$$

Then the category of elements of S would look like the (discrete) category at the top:

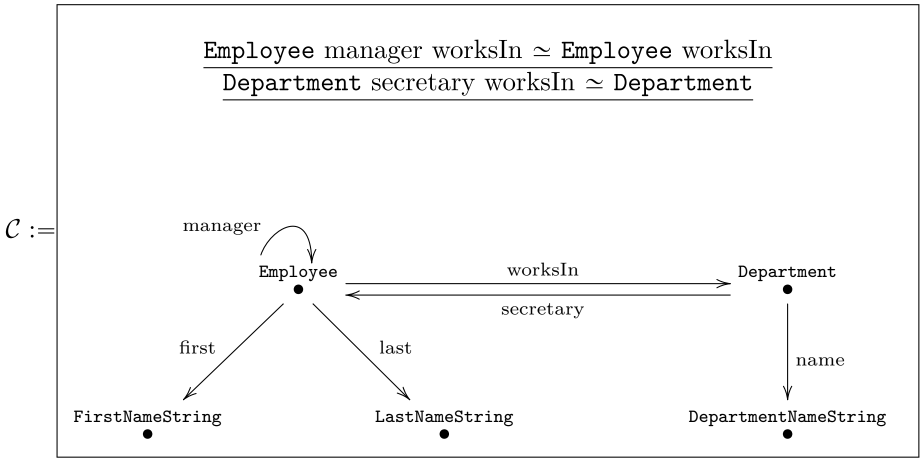


(4.25)

We also see that the category of elements construction has converted an A-indexed set into a relative set over A, as in Definition 2.7.6.7.

The above example does not show at all how the Grothendieck construction transforms a database instance into an RDF triple store. The reason is that our database schema was A, a discrete category that specifies no connections between data (it simply collects the data into bins). So lets examine a more interesting database schema and instance. This is taken from [Sp2].

Application 4.6.2.3. Consider the schema below, which we first encountered in Example 3.5.2.1:



(4.26)

And consider the instance $J\colon \mathcal{C} \to \mathbf{Set}$, which we first encountered in (3.13) and (3.15)

Employee				
ID	**first**	**last**	**manager**	**worksIn**
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Emmy	Noether	103	q10

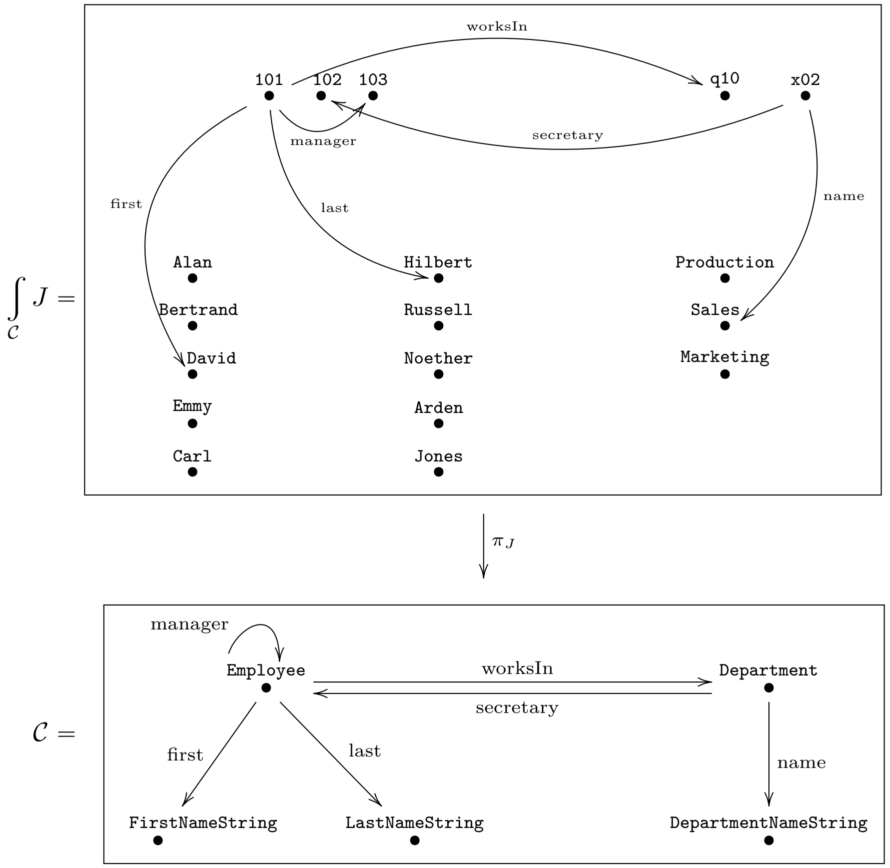
Department		
ID	**name**	**secretary**
q10	Sales	101
x02	Production	102

FirstNameString
ID
Alan
Bertrand
Carl
David
Emmy

LastNameString
ID
Arden
Hilbert
Jones
Noether
Russell

DepartmentNameString
ID
Marketing
Production
Sales

The category of elements of $J\colon \mathcal{C} \to \mathbf{Set}$ looks like this:



(4.27)

In the above drawing (4.27) of $\int_{\mathcal{C}} J$, we left out 10 arrows for ease of readability, for example, we left out an arrow $\overset{\texttt{102}}{\bullet}\xrightarrow{\texttt{first}}\overset{\texttt{Bertrand}}{\bullet}$.

For the punchline, how do we see the category of elements $\int_{\mathcal{C}} J$ as an RDF triple store? For each arrow in $\int_{\mathcal{C}} J$, we take the triple consisting of the source vertex, the arrow name, and the target vertex. So our triple store would include triples such as $\langle$`102 first Bertrand`$\rangle$ and $\langle$`101 manager 103`$\rangle$.

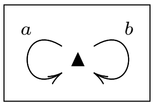
◊◊

Exercise 4.6.2.4. Come up with a schema and instance whose category of elements contains (at least) the data from (4.24). ◊

Slogan 4.6.2.5.

> “ *The Grothendieck construction takes structured, boxed-up data and flattens it by throwing it all into one big space. The projection functor is then tasked with remembering which box each datum originally came from.* ”

Exercise 4.6.2.6. Recall from Section 3.1.2.10 that a finite state machine is a free monoid $(\text{List}(\Sigma), [\ ], +\!\!+)$ acting on a set X. Recall also that we can consider a monoid as a category $\mathcal{M}$ with one object and a monoid action as a set-valued functor $F\colon \mathcal{M} \to \mathbf{Set}$, (see Section 4.2.1.1). In the case of Figure 3.1 the monoid in question is $\text{List}(a, b)$, which can be drawn as the schema



and the functor $F\colon \mathcal{M} \to \mathbf{Set}$ is recorded in an action table in Example 3.1.3.1. What is $\int_{\mathcal{M}} F$? How does it relate to the picture in Figure 3.1? ◊

4.6.3 Full subcategory

Definition 4.6.3.1. Let $\mathcal{C}$ be a category and let $X \subseteq \text{Ob}(\mathcal{C})$ be a set of objects in $\mathcal{C}$. The *full subcategory of* $\mathcal{C}$ *spanned by* X is the category, which we denote by $\mathcal{C}_{\text{Ob}=X}$, with objects $\text{Ob}(\mathcal{C}_{\text{Ob}=X}) := X$ and with morphisms $\text{Hom}_{\mathcal{C}_{\text{Ob}=X}}(x, x') := \text{Hom}_{\mathcal{C}}(x, x')$.

Example 4.6.3.2. The following are examples of full subcategories. We will name them in the form “X inside of Y”, and each time we mean that X and Y are names of categories, the category X can be considered as a subcategory of the category Y in some sense, and it is full. In other words, all morphisms in Y “count” as morphisms in X.

- Finite sets inside of sets, $\mathbf{Fin} \subseteq \mathbf{Set}$;
- Finite sets of the form $\underline{n}$ inside of $\mathbf{Fin}$;
- Linear orders of the form $[n]$ inside of all finite linear orders, $\mathbf{\Delta} \subseteq \mathbf{FLin}$;
- Groups inside of monoids, $\mathbf{Grp} \subseteq \mathbf{Mon}$;
- Monoids inside of categories, $\mathbf{Mon} \subseteq \mathbf{Cat}$;
- Sets inside of graphs, $\mathbf{Set} \subseteq \mathbf{Grph}$;
- Partial orders (resp. linear orders) inside of $\mathbf{PrO}$;

- Discrete categories (resp. indiscrete categories) inside of **Cat**;

Remark 4.6.3.3. A subcategory $\mathcal{C} \subseteq \mathcal{D}$ is (up to isomorphism) just a functor $i\colon \mathcal{C} \to \mathcal{D}$ that happens to be injective on objects and arrows. The subcategory is full if and only if i is a full functor in the sense of Definition 4.3.4.12.

Example 4.6.3.4. Let $\mathcal{C}$ be a category, let $X \subseteq \mathrm{Ob}(\mathcal{C})$ be a set of objects, and let $\mathcal{C}_{\mathrm{Ob}=X}$ denote the full subcategory of $\mathcal{C}$ spanned by X. We can realize this as a fiber product of categories. Indeed, recall that for any set, we can form the indiscrete category on that set; see Example 4.3.4.3. In fact, we have a functor $Ind\colon \mathbf{Set} \to \mathbf{Cat}$. Thus our function $X \to \mathrm{Ob}(\mathcal{C})$ can be converted into a functor between indiscrete categories $Ind(X) \to Ind(\mathrm{Ob}(\mathcal{C}))$. There is also a functor $\mathcal{C} \to Ind(\mathrm{Ob}(\mathcal{C}))$ sending each object to itself. Then the full subcategory of $\mathcal{C}$ spanned by X is the fiber product of categories,

$$\begin{tikzcd}
\mathcal{C}_{\mathrm{Ob}=X} \arrow[r] \arrow[d] & \mathcal{C} \arrow[d] \\
Ind(X) \arrow[r] & Ind(\mathrm{Ob}(\mathcal{C}))
\end{tikzcd}$$

Exercise 4.6.3.5. Including all identities and all compositions, how many morphisms are there in the full subcategory of **Set** spanned by the objects $\{\underline{0}, \underline{1}, \underline{2}\}$? Write them out. ◊

4.6.4 Comma categories

Category theory includes a highly developed and interoperable catalogue of materials and production techniques. One such is the comma category.

Definition 4.6.4.1. Let $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ be categories and let $F\colon \mathcal{A} \to \mathcal{C}$ and $G\colon \mathcal{B} \to \mathcal{C}$ be functors. The *comma category of $\mathcal{C}$ morphisms from F to G*, denoted $(F \downarrow_{\mathcal{C}} G)$ or simply $(F \downarrow G)$, is the category with objects

$$\mathrm{Ob}(F \downarrow G) = \{(a, b, f) \mid a \in \mathrm{Ob}(\mathcal{A}), b \in \mathrm{Ob}(\mathcal{B}), f\colon F(a) \to G(b) \text{ in } \mathcal{C}\}$$

and for any two objects (a, b, f) and (a', b', f') the set $\mathrm{Hom}_{(F\downarrow G)}((a, b, f), (a', b', f'))$ of morphisms $(a, b, f) \longrightarrow (a', b', f')$ is

$$\{(q, r) \mid q\colon a \to a' \text{ in } \mathcal{A},\ \ r\colon b \to b' \text{ in } \mathcal{B},\ \text{such that } f' \circ F(q) = G(r) \circ f\}.$$

In pictures,

$$\mathrm{Hom}_{(F\downarrow G)}((a, b, f), (a', b', f')) := \left\{ \begin{tikzcd}
a \arrow[d, "q"'] & F(a) \arrow[r, "f"] \arrow[d, "F(q)"'] \arrow[dr, phantom, "\checkmark"] & G(b) \arrow[d, "G(r)"] & b \arrow[d, "r"] \\
a' & F(a') \arrow[r, "f'"'] & G(b') & b'
\end{tikzcd} \right\}$$

We refer to the diagram $\mathcal{A} \xrightarrow{F} \mathcal{C} \xleftarrow{G} \mathcal{B}$ (in **Cat**) as the *setup* for the comma category $(F \downarrow G)$.

There is a canonical functor $(F \downarrow G) \to \mathcal{A}$ called *left projecton*, sending (a, b, f) to a, and a canonical functor $(F \downarrow G) \to \mathcal{B}$ called *right projection*, sending (a, b, f) to b.

A setup $\mathcal{A} \xrightarrow{F} \mathcal{C} \xleftarrow{G} \mathcal{B}$ is reversable; i.e. we can flip it to obtain $\mathcal{B} \xrightarrow{G} \mathcal{C} \xleftarrow{F} \mathcal{A}$. However, note that $(F \downarrow G)$ is different than (i.e. almost never equivalent to) $(G \downarrow F)$, unless every arrow in $\mathcal{C}$ is an isomorphism.

Slogan 4.6.4.2.

> “ *When two categories $\mathcal{A}, \mathcal{B}$ can be interpreted in a common setting $\mathcal{C}$, the comma category integrates them by recording how to move from $\mathcal{A}$ to $\mathcal{B}$ inside $\mathcal{C}$.* ”

Example 4.6.4.3. Let $\mathcal{C}$ be a category and $I\colon \mathcal{C} \to \mathbf{Set}$ a functor. In this example we show that the comma category construction captures the notion of taking the category of elements $\int_{\mathcal{C}} I$; see Definition 4.6.2.1.

Consider the set $\underline{1}$, the category $Disc(\underline{1})$, and the functor $F\colon Disc(\underline{1}) \to \mathbf{Set}$ sending the unique object to the set $\underline{1}$. We use the comma category setup $\underline{1} \xrightarrow{F} \mathbf{Set} \xleftarrow{I} \mathcal{C}$. There is an isomorphism of categories

$$\int_{\mathcal{C}} I \cong (F \downarrow I).$$

Indeed, an object in $(F \downarrow I)$ is a triple (a, b, f) where $a \in \text{Ob}(\underline{1}), b \in \text{Ob}(\mathcal{C})$, and $f\colon F(a) \to I(b)$ is a morphism in **Set**. There is only one object in $\underline{1}$, so this reduces to a pair (b, f) where $b \in \text{Ob}(\mathcal{C})$ and $f\colon \{☺\} \to I(b)$. The set of functions $\{☺\} \to I(b)$ is isomorphic to $I(b)$, as we saw in Exercise 2.1.2.14. So we have reduced $\text{Ob}(F \downarrow I)$ to the set of pairs (b, x) where $b \in \text{Ob}(\mathcal{C})$ and $x \in I(b)$; this is $\text{Ob}(\int_{\mathcal{C}} I)$. Because there is only one function $\underline{1} \to \underline{1}$, a morphism $(b, x) \to (b', x')$ in $(F \downarrow I)$ boils down to a morphism $r\colon b \to b'$ such that the diagram

$$\begin{tikzcd}
\underline{1} \arrow[r, "x"] \arrow[d, equal] & I(b) \arrow[d, "I(r)"] \\
\underline{1} \arrow[r, "x'"'] & I(b')
\end{tikzcd}$$

commutes. But such diagrams are in one-to-one correspondence with the diagrams needed for morphisms in $\int_{\mathcal{C}} I$.

Exercise 4.6.4.4. Let $\mathcal{C}$ be a category and let $c, c' \in \text{Ob}(\mathcal{C})$ be objects. Consider them as functors $c, c'\colon \underline{1} \to \mathcal{C}$, and consider the setup $\underline{1} \xrightarrow{c} \mathcal{C} \xleftarrow{c'} \underline{1}$. What is the comma category $(c \downarrow c')$? ◊

4.6.5 Arithmetic of categories

In Section 2.7.3, we summarized some of the properties of products, coproducts, and exponentials for sets, attempting to show that they lined up precisely with familiar arithmetic properties of natural numbers. Astoundingly, we can do the same for categories.

In the following proposition, we denote the coproduct of two categories $\mathcal{A}$ and $\mathcal{B}$ by the notation $\mathcal{A} + \mathcal{B}$ rather than $\mathcal{A} \sqcup \mathcal{B}$. We also denote the functor category $\text{Fun}(\mathcal{A}, \mathcal{B})$ by $\mathcal{B}^{\mathcal{A}}$. Finally, we use $\underline{0}$ and $\underline{1}$ to refer to the discrete category on 0 and on 1 object, respectively.

Proposition 4.6.5.1. *The following isomorphisms exist for any small categories $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$.*

- $\mathcal{A} + \underline{0} \cong \mathcal{A}$
- $\mathcal{A} + \mathcal{B} \cong \mathcal{B} + \mathcal{A}$
- $(\mathcal{A} + \mathcal{B}) + \mathcal{C} \cong \mathcal{A} + (\mathcal{B} + \mathcal{C})$
- $\mathcal{A} \times \underline{0} \cong \underline{0}$
- $\mathcal{A} \times \underline{1} \cong \mathcal{A}$
- $\mathcal{A} \times \mathcal{B} \cong \mathcal{B} \times \mathcal{A}$
- $(\mathcal{A} \times \mathcal{B}) \times \mathcal{C} \cong \mathcal{A} \times (\mathcal{B} \times \mathcal{C})$
- $\mathcal{A} \times (\mathcal{B} + \mathcal{C}) \cong (\mathcal{A} \times \mathcal{B}) + (\mathcal{A} \times \mathcal{C})$
- $\mathcal{A}^{\underline{0}} \cong \underline{1}$
- $\mathcal{A}^{\underline{1}} \cong \mathcal{A}$
- $\underline{0}^{\mathcal{A}} \cong \underline{0}, \quad \textit{if } \mathcal{A} \neq \underline{0}$
- $\underline{1}^{\mathcal{A}} \cong \underline{1}$
- $\mathcal{A}^{\mathcal{B}+\mathcal{C}} \cong \mathcal{A}^{\mathcal{B}} \times \mathcal{A}^{\mathcal{C}}$
- $(\mathcal{A}^{\mathcal{B}})^{\mathcal{C}} \cong \mathcal{A}^{\mathcal{B} \times \mathcal{C}}$

Proof. These are standard results; see [Mac].

□

Chapter 5

Categories at work

We have now set up an understanding of the basic notions of category theory: categories, functors, natural transformations, and universal properties. We have discussed many sources of examples: orders, graphs, monoids, and databases. We begin this chapter with the notion of *adjoint functors* (also known as *adjunctions*), which are like dictionaries that translate back and forth between different categories.

5.1 Adjoint functors

Just above, in the introduction to this chapter, I said that adjoint functors are like dictionaries that translate back and forth between different categories. How far can we take that analogy?

In the common understanding of dictionaries, we assume that the two languages (say French and English) are equally expressive, and that a good dictionary will be an even exchange of ideas. But in category theory we often have two categories that are not on the same conceptual level. This is most clear in the case of so-called *free-forgetful adjunctions*. In Section 5.1.1 we will explore the sense in which each adjunction provides a dictionary between two categories that are not necessarily on an equal footing, so to speak.

5.1.1 Discussion and definition

Consider the category of monoids and the category of sets. A monoid $(M, e, \star)$ is a set with an identity element and a multiplication formula that is associative. A set is just a set. A dictionary between **Mon** and **Set** should not be required to set up an even exchange, but instead an exchange that is appropriate to the structures at hand. It will be in the form of two functors, one we'll denote by $L\colon \mathbf{Set} \to \mathbf{Mon}$, and one we'll denote by $R\colon \mathbf{Mon} \to \mathbf{Set}$. But to say what "appropriate" means requires more work.

Let's bring it down to earth with an analogy. A one-year-old can make repeatable noises and an adult can make repeatable noises. One might say "after all, talking is nothing but making repeatable noises." But the adult's repeatable noises are called words, they form sentences, and these sentences can cause nuclear wars. There is something more in adult language than there is simply in repeatable sounds. In the same vein, a tennis match can be viewed as physics, but you won't see the match. So we have something analogous to two categories here: ((repeated noises)) and ((meaningful words)).

We are looking for adjoint functors going back and forth, serving as the appropriate sort of dictionary.

To translate baby talk into adult language we would make every repeated noise a kind of word, thereby granting it meaning. We don't know what a given repeated noise should mean, but we give it a slot in our conceptual space, always pondering "I wonder what she means by Konnen.." On the other hand, to translate from meaningful words to repeatable noises is easy. We just hear the word as a repeated noise, which is how the baby probably hears it.

Adjoint functors often come in the form of "free" and "forgetful". Here we freely add Konnen to our conceptual space without having any idea how it adheres to the rest of the child's noises or feelings. But it doesn't act like a sound to us, it acts like a word; we don't know what it means but we figure it means something. Conversely, the translation going the other way is "forgetful", forgetting the meaning of our words and just hearing them as sounds. The baby hears our words and accepts them as mere sounds, not knowing that there is anything extra to get.

Back to sets and monoids, the sets are like the babies from our story: they are simple objects full of unconnected dots. The monoids are like adults, forming words and performing actions. In the monoid, each element means something and combines with other elements in some way. There are lots of different sets and lots of different monoids, just as there are many babies and many adults, but there are patterns to the behavior of each kind and we put them in different categories.

Applying free functor $L\colon \mathbf{Set} \to \mathbf{Mon}$ to a set X makes every element $x \in X$ a word, and these words can be strung together to form more complex words. (We discussed the free functor in Section 3.1.1.12.) Since a set such as X carries no information about the meaning or structure of its various elements, the free monoid $F(X)$ does not relate different words in any way. To apply the forgetful functor $R\colon \mathbf{Mon} \to \mathbf{Set}$ to a monoid, even a structured one, is to simply forget that its elements are anything but mere elements of a set. It sends a monoid $(M, 1, \star)$ to the set M.

The analogy is complete. However, this is all just ideas. Let's give a definition, then return to our sets, monoids, sounds, and words.

Definition 5.1.1.1. Let $\mathcal{B}$ and $\mathcal{A}$ be categories. [1] An *adjunction between $\mathcal{B}$ and $\mathcal{A}$* is a pair of functors

$$L\colon \mathcal{B} \to \mathcal{A} \qquad \text{and} \qquad R\colon \mathcal{A} \to \mathcal{B}$$

together with a natural isomorphism [2] whose component for any objects $A \in \mathrm{Ob}(\mathcal{A})$ and $B \in \mathrm{Ob}(\mathcal{B})$ is:

$$\alpha_{B,A}\colon \mathrm{Hom}_{\mathcal{A}}(L(B), A) \xrightarrow{\cong} \mathrm{Hom}_{\mathcal{B}}(B, R(A)). \tag{5.1}$$

This isomorphism is called the *adjunction isomorphism* for the (L, R) adjunction, and for any morphism $f\colon L(B) \to A$ in $\mathcal{A}$, we refer to $\alpha_{B,A}(f)\colon B \to R(A)$ as *the adjunct* of f. [3]

[1] Throughout this definition, notice that B's come before A's, especially in (5.1), which might be confusing. It was a stylistic choice to match with the **B**abies and **A**dults discussion above and below this definition.

[2] The natural isomorphism α (see Lemma 4.3.2.12) is between two functors $\mathcal{B}^{\mathrm{op}} \times \mathcal{A} \to \mathbf{Set}$, namely the functor $(B, A) \mapsto \mathrm{Hom}_{\mathcal{A}}(L(B), A)$ and the functor $(B, A) \mapsto \mathrm{Hom}_{\mathcal{B}}(B, R(A))$.

[3] Conversely, for any $g\colon B \to R(A)$ in $\mathcal{B}$ we refer to $\alpha_{B,A}^{-1}(g)\colon L(B) \to A$ as *the adjunct* of g.

The functor L is called the *left adjoint* and the functor R is called the *right adjoint.* We may say that *L is the left adjoint of R* or that *R is the right adjoint of L.* [4] We often denote this setup by

$$L\colon \mathcal{B} \rightleftarrows \mathcal{A} :R$$

Proposition 5.1.1.2. *Let $L\colon \mathbf{Set} \to \mathbf{Mon}$ be the functor sending $X \in \mathrm{Ob}(\mathbf{Set})$ to the free monoid $L(X) := (\mathrm{List}(X), [\ ], +\!\!+)$, as in Definition 3.1.1.15. Let $R\colon \mathbf{Mon} \to \mathbf{Set}$ be the functor sending each monoid $\mathcal{M} := (M, 1, \star)$ to its underlying set $R(\mathcal{M}) := M$. Then L is left adjoint to R.*

Proof. If we can find a natural isomorphism of sets

$$\alpha_{X,\mathcal{M}}\colon \mathrm{Hom}_{\mathbf{Mon}}(L(X), \mathcal{M}) \to \mathrm{Hom}_{\mathbf{Set}}(X, R(\mathcal{M}))$$

we will have succeeded in showing that these functors are adjoint.

Suppose given an element $f \in \mathrm{Hom}_{\mathbf{Mon}}(L(X), \mathcal{M})$, i.e. a monoid homomorphism $f\colon \mathrm{List}(X) \to M$ (sending $[\ ]$ to 1 and list concatenation to $\star$). Then in particular we can apply f to the singleton list $[x]$ for any $x \in X$. This gives a function $X \to M$ by $x \mapsto f([x])$, and this is $\alpha_{X,\mathcal{M}}(f)\colon X \to M = R(\mathcal{M})$. We need only to supply an inverse $\beta_{X,\mathcal{M}}\colon \mathrm{Hom}_{\mathbf{Set}}(X, R(\mathcal{M})) \to \mathrm{Hom}_{\mathbf{Mon}}(L(X), \mathcal{M})$.

Suppose given an element $g \in \mathrm{Hom}_{\mathbf{Set}}(X, R(\mathcal{M}))$, i.e. a function $g\colon X \to M$. Then to any list $\ell = [x_1, x_2, \ldots, x_n] \in \mathrm{List}(X)$ we can assign $\beta_{X,\mathcal{M}}(\ell) := g(x_1) \star g(x_2) \star \cdots \star g(x_n)$ (if $\ell = [\ ]$ is the empty list, assign $\beta_{X,\mathcal{M}}([\ ]) := 1$). We now have a function $\mathrm{List}(X) \to M$. It is a monoid homomorphism because it respects identity and composition. It is easy to check that β and α are mutually inverse, completing the proof.

□

Example 5.1.1.3. We need to ground our discussion in some concrete mathematics. In Proposition 5.1.1.2 we provided our long-awaited adjunction between sets and monoids. A set X gets transformed into a monoid by considering lists in X; a monoid $\mathcal{M}$ gets transformed into a set by forgetting the multiplication law. So we have a functor going one way and the other,

$$L\colon \mathbf{Set} \to \mathbf{Mon}, \qquad\qquad R\colon \mathbf{Mon} \to \mathbf{Set},$$

but an adjunction is more than that: it includes a guarantee about the relationship between these two functors. What is the relationship between L and R? Consider an arbitrary monoid $\mathcal{M} = (M, 1, *)$.

If I want to pick out 3 elements of the set M, that's the same thing as giving a function $\{a, b, c\} \to M$. But that function exists in the category of sets; in fact it is an element of $\mathrm{Hom}_{\mathbf{Set}}(\{a, b, c\}, M)$. But since $M = R(\mathcal{M})$ is the underlying set of our monoid, we can view the current paragraph in the light of our adjunction Equation (5.1) by saying it has been about the set

$$\mathrm{Hom}_{\mathbf{Set}}(\{a, b, c\}, R(\mathcal{M})).$$

This set classifies all the ways to pick three elements out of the underlying set of our monoid $\mathcal{M}$. It was constructed completely from within the category **Set**.

[4] The left adjoint does not have to be called L, nor does the right adjoint have to be called R, of course. This is suggestive.

Now we ask what Equation (5.1) means. The equation

$$\mathrm{Hom}_{\mathbf{Mon}}(L(\{a,b,c\}),\mathcal{M}) \cong \mathrm{Hom}_{\mathbf{Set}}(\{a,b,c\},R(\mathcal{M})).$$

tells us that somehow we can answer the same question completely from within the category of monoids. In fact it tells us how to do so, namely as $\mathrm{Hom}_{\mathbf{Mon}}(\mathrm{List}(\{1,2,3\},\mathcal{M})$. Exercise 5.1.1.4 looks at how that should go. The answer is "hidden" in the proof of Proposition 5.1.1.2.

Exercise 5.1.1.4. Let $X = \{a,b,c\}$ and let $\mathcal{M} = (\mathbb{N},1,*)$ be the multiplicative monoid of natural numbers (see Example 3.1.3.2). Let $f\colon X \to \mathbb{N}$ be the function given by $f(a) = 7, f(b) = 2, f(c) = 2$, and let $\beta_{X,\mathcal{M}}\colon \mathrm{Hom}_{\mathbf{Set}}(X,R(\mathcal{M})) \to \mathrm{Hom}_{\mathbf{Mon}}(L(X),\mathcal{M})$ be as in the proof of Proposition 5.1.1.2. What is $\beta_{X,\mathcal{M}}(f)([b,b,a,c])$? ◊

Let us look once more at the adjunction between adults and babies. Using the notation of Definition 5.1.1.1 $\mathcal{A}$ is the "adult category" of meaningful words and $\mathcal{B}$ is the "baby category" of repeated noises. The left adjoint turns every repeated sound into a meaningful word (having "free" meaning) and the right adjoint "forgets" the meaning of any word and considers it merely as a sound.

At the risk of taking this simple analogy too far, let's have a go at the heart of the issue: how to conceive of the isomorphism (5.1) of Hom's. Once we have freely given a slot to each of baby's repeated sounds, we try to find a mapping from the lexicon $L(B)$ of these new words to our own lexicon A of meaningful words; these are mappings in the adult category $\mathcal{A}$ of the form $L(B) \to A$. And (stretching it) the baby tries to find a mapping (which we might see as emulation) from her set B of repeatable sounds to the set $R(A)$ of the sounds the adult seems to repeat. If there was a global system for making these transformations that would establish (5.1) and hence the adjunction.

Note that the directionality of the adjunction makes a difference. If $L\colon \mathcal{B} \to \mathcal{A}$ is left adjoint to $R\colon \mathcal{A} \to \mathcal{B}$ we rarely have an isomorphism $\mathrm{Hom}_{\mathcal{A}}(A,L(B)) \cong \mathrm{Hom}_{\mathcal{B}}(R(A),B)$. In the case of babies and adults, we see that it would make little sense to look for a mapping in the category of meaningful words from the adult lexicon to the wordifications of baby-sounds, because there is unlikely to be a good candidate for most of our words. That is, to which of our child's repeated noises would we assign the concept "weekday"?

Again, the above is simply an analogy, and almost certainly not formalizable. The next example shows mathematically the point we tried to make in the previous paragraph, that the directionality of an adjunction is not arbitrary.

Example 5.1.1.5. Let $L\colon \mathbf{Set} \to \mathbf{Mon}$ and $R\colon \mathbf{Mon} \to \mathbf{Set}$ be the free and forgetful functors from Proposition 5.1.1.2. We know that L is left adjoint to R; however L is *not* right adjoint to R. In other words, we can show that the necessary natural isomorphism cannot exist.

Let $X = \{a,b\}$ and let $\mathcal{M} = (\{1\},1,!)$ be the trivial monoid. Then the necessary natural isomorphism would need to give us a bijection

$$\mathrm{Hom}_{\mathbf{Mon}}(\mathcal{M},L(X)) \cong^{?} \mathrm{Hom}_{\mathbf{Set}}(\{1\},X).$$

But the left-hand side has one element, because $\mathcal{M}$ is the initial object in $\mathbf{Mon}$ (see Example 4.5.3.8), whereas the right-hand side has two elements. Therefore no isomorphism can exist.

Example 5.1.1.6. Preorders have underlying sets, giving rise to a functor $U\colon \mathbf{PrO} \to \mathbf{Set}$. The functor U has both a left adjoint and a right adjoint. The left adjoint of U is $D\colon \mathbf{Set} \to \mathbf{PrO}$, sending a set X to the discrete preorder on X (the preorder with underlying set X, having the fewest possible $\leqslant$'s). The right adjoint of U is $I\colon \mathbf{Set} \to \mathbf{PrO}$, sending a set X to the indiscrete preorder on X (the preorder with underlying set X, having the most possible $\leqslant$'s). See Example 3.4.4.5.

Exercise 5.1.1.7. Let $U\colon \mathbf{Grph} \to \mathbf{Set}$ denote the functor sending a graph to its underlying set of vertices. This functor has both a left and a right adjoint.

a.) What functor $\mathbf{Set} \to \mathbf{Grph}$ is the left adjoint of U?

b.) What functor $\mathbf{Set} \to \mathbf{Grph}$ is the right adjoint of U?

◊

Example 5.1.1.8. Here are some other adjunctions:

- $\mathrm{Ob}\colon \mathbf{Cat} \to \mathbf{Set}$ has a left adjoint $\mathbf{Set} \to \mathbf{Cat}$ given by the discrete category.
- $\mathrm{Ob}\colon \mathbf{Cat} \to \mathbf{Set}$ has a right adjoint $\mathbf{Set} \to \mathbf{Cat}$ given by the indiscrete category.
- The underlying graph functor $\mathbf{Cat} \to \mathbf{Grph}$ has a left adjoint $\mathbf{Grph} \to \mathbf{Cat}$ given by the free category.
- The functor $\mathbf{PrO} \to \mathbf{Grph}$, given by drawing edges for $\leqslant$'s, has a left adjoint given by existence of paths.
- The forgetful functor from posets to preorders has a left adjoint given by quotient by isomorphism relation.
- Given a set A, the functor $(- \times A)\colon \mathbf{Set} \to \mathbf{Set}$ has a right adjoint $\mathrm{Hom}(A, -)$ (this was called currying in Section 2.7.2).

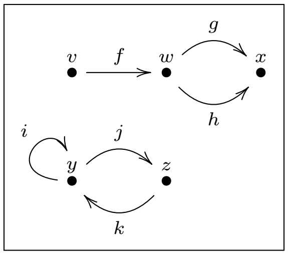
Exercise 5.1.1.9. Let $F\colon \mathcal{C} \to \mathcal{D}$ and $G\colon \mathcal{D} \to \mathcal{C}$ be mutually inverse equivalences of categories (see Definition 4.3.4.1). Are they adjoint in one direction or the other? ◊

Exercise 5.1.1.10. The discrete category functor $Disc\colon \mathbf{Set} \to \mathbf{Cat}$ has a left adjoint $p\colon \mathbf{Cat} \to \mathbf{Set}$.

a.) For an arbitrary object $X \in \mathrm{Ob}(\mathbf{Set})$ and an arbitrary object $\mathcal{C} \in \mathrm{Ob}(\mathbf{Cat})$, write down the adjunction isomorphism.

b.) Let $\mathcal{C}$ be the free category on the graph G:

$G :=$



and let $X = \{1, 2, 3\}$. How many elements does the set $\mathrm{Hom}_{\mathbf{Set}}(\mathcal{C}, Disc(X))$ have?

c.) What can you do to an arbitrary category $\mathcal{C}$ to make a set $p(\mathcal{C})$ such that the adjunction isomorphism holds? That is, how does the functor p behave on objects?

◊

The following proposition says that all adjoints to a given functor are isomorphic to each other.

Proposition 5.1.1.11. *Let $\mathcal{C}$ and $\mathcal{D}$ be categories, let $F\colon \mathcal{C} \to \mathcal{D}$ be a functor, and let $G, G'\colon \mathcal{D} \to \mathcal{C}$ also be functors. If both G and G' are right adjoint (respectively left adjoint) to F then there is a natural isomorphism $\phi\colon G \to G'$.*

Proof. Suppose that both G and G' are right adjoint to F (the case of G and G' being left adjoint is similarly proved). We first give a formula for the components of $\phi\colon G \to G'$ and its inverse $\psi\colon G' \to G$. Given an object $d \in \mathrm{Ob}(\mathcal{D})$, we use $c = G(d)$ to obtain two natural isomorphisms, one from each adjunction:

$$\mathrm{Hom}_{\mathcal{C}}(G(d), G(d)) \cong \mathrm{Hom}_{\mathcal{D}}(F(G(d)), d) \cong \mathrm{Hom}_{\mathcal{C}}(G(d), G'(d)).$$

The identity component $\mathrm{id}_{G(d)}$ is then sent to some morphism $G(d) \to G'(d)$, which we take to be ϕ_d. Similarly, we use $c' = G'(d)$ to obtain two natural isomorphisms, one from each adjunction:

$$\mathrm{Hom}_{\mathcal{C}}(G'(d), G'(d)) \cong \mathrm{Hom}_{\mathcal{D}}(F(G'(d)), d) \cong \mathrm{Hom}_{\mathcal{C}}(G'(d), G(d)).$$

Again, the identity component $\mathrm{id}_{G'(d)}$ is sent to some morphism $G'(d) \to G(d)$, which we take to be ψ_d. The naturality of the isomorphisms implies that ϕ and ψ are natural transformations, and it is straightforward to check that they are mutually inverse.

□

5.1.1.12 Quantifiers as adjoints

One of the simplest but neatest places that adjoints show up is between preimages and the logical quantifiers $\exists$ and $\forall$, which we first discussed in Notation 2.1.1.1. The setting in which to discuss this is that of sets and their power preorders. That is, if X is a set then recall from Section 3.4.2 that the power set $\mathbb{P}(X)$ has a natural ordering by inclusion of subsets.

Given a function $f\colon X \to Y$ and a subset $V \subseteq Y$ the preimage is $f^{-1}(V) := \{x \in X \mid f(x) \in V\}$. If $V' \subseteq V$ then $f^{-1}(V') \subseteq f^{-1}(V)$, so in fact $f^{-1}\colon \mathbb{P}(Y) \to \mathbb{P}(X)$ can be considered a functor (where of course we are thinking of preorders as categories). The quantifiers appear as adjoints of f^{-1}.

Let's begin with the left adjoint of $f^{-1}\colon \mathbb{P}(Y) \to \mathbb{P}(X)$. It is a functor $L_f\colon \mathbb{P}(X) \to \mathbb{P}(Y)$. Choose an object $U \subseteq X$ in $\mathbb{P}(X)$. It turns out that

$$L_f(U) = \{y \in Y \mid \exists x \in f^{-1}(y) \text{ such that } x \in U\}.$$

And the right adjoint $R_f\colon \mathbb{P}(X) \to \mathbb{P}(Y)$, when applied to U is

$$R_f(U) = \{y \in Y \mid \forall x \in f^{-1}(y), x \in U\}.$$

In fact, the functor L_f is generally denoted $\exists_f\colon \mathbb{P}(X) \to \mathbb{P}(Y)$, and R_f is generally denoted $\forall_f\colon \mathbb{P}(X) \to \mathbb{P}(Y)$.

$$\begin{tikzcd}
\mathbb{P}(X) \arrow[r, bend left, "\exists_f"] \arrow[r, bend right, "\forall_f"] & \mathbb{P}(Y). \arrow[l, "f^{-1}"']
\end{tikzcd}$$

We will see in the next example why this notation is apt.

Example 5.1.1.13. In logic or computer science, the quantifiers $\exists$ and $\forall$ are used to ask whether any or all elements of a set have a certain property. For example, one may have a set of natural numbers and want to know whether any or all are even or odd. Let $Y = \{\texttt{even}, \texttt{odd}\}$, and let $p\colon \mathbb{N} \to Y$ be the function that takes assigns to each natural number its parity (even or odd). Because the elements of $\mathbb{P}(\mathbb{N})$ and $\mathbb{P}(Y)$ are ordered by "inclusion of subsets", we can construe these orders as categories (by Proposition 4.2.1.17). That's all old; what's new is that we have adjunctions between these categories

$$\begin{tikzcd}
\mathbb{P}(\mathbb{N}) \arrow[r, bend left, "\exists_p"] \arrow[r, bend right, "\forall_p"] & \mathbb{P}(Y). \arrow[l, "p^{-1}"']
\end{tikzcd}$$

Given a subset $U \subseteq \mathbb{N}$, i.e. an object $U \in \mathrm{Ob}(\mathbb{P}(\mathbb{N}))$, we investigate the objects $\exists_p(U), \forall_p(U)$. These are both subsets of $\{\texttt{even}, \texttt{odd}\}$. The set $\exists_p(U)$ includes the element `even` if there exists an even number in U; it includes the element `odd` if there exists an odd number in U. Similarly, the set $\forall_p(U)$ includes the element `even` if every even number is in U and it includes `odd` if every odd number is in U. [5]

We explain just one of these in terms of the definitions. Let $V = \{\texttt{even}\} \subseteq Y$. Then $f^{-1}(V) \subseteq \mathbb{N}$ is the set of even numbers, and there is a morphism $f^{-1}(V) \to U$ in $\mathbb{P}(\mathbb{N})$ if and only if U contains all the even numbers. Therefore, the adjunction isomorphism $\mathrm{Hom}_{\mathbb{P}(\mathbb{N})}(f^{-1}(V), U) \cong \mathrm{Hom}_{\mathbb{P}(Y)}(V, \forall_p U)$ says that $V \subseteq \forall_p U$, i.e. $\forall_p(U)$ includes the element `even` if and only if U contains all the even numbers, as we said above.

Exercise 5.1.1.14. The national Scout jamboree is a gathering of Boy Scouts from troops across the US. Let X be the set of Boy Scouts in the US, and let Y be the set of Boy Scout troops in the US. Let $t\colon X \to Y$ be the function that assigns to each Boy Scout his troop. Let $U \subseteq X$ be the set of Boy Scouts in attendance at this years jamboree. What is the meaning of the objects $\exists_t U$ and $\forall_t U$? ◊

Exercise 5.1.1.15. Let X be a set and $U \subseteq X$ a subset. Find a set Y and a function $f\colon X \to Y$ such that $\exists_f(U)$ somehow tells you whether U is non-empty, and such that $\forall_f(U)$ somehow tells you whether $U = X$. ◊

In fact, "quantifiers as adjoints" is part of a larger story. Suppose we think of elements of a set X as bins, or storage areas. An element of $\mathbb{P}(X)$ can be construed as an injection $U \hookrightarrow X$, i.e. an assignment of a bin to each element of U, with at most one element of U in each bin. Relaxing that restriction, we may consider arbitrary sets U and assignments $U \to X$ of a bin to each element $u \in U$. Given a function $f\colon X \to Y$, we can generalize $\exists_f$ and $\forall_f$ to functors denoted Σ_f and Π_f, which will parameterize disjoint unions and products (respectively) over $y \in Y$. This will be discussed in Section 5.1.4.

5.1.2 Universal concepts in terms of adjoints

In this section we discuss how universal concepts, i.e. initial objects and terminal objects, colimits and limits, are easily phrased in the language of adjoint functors. We will say that a functor $F\colon \mathcal{C} \to \mathcal{D}$ *is a left adjoint* if there exists a functor $G\colon \mathcal{D} \to \mathcal{C}$ such that F is a left adjoint of G. We showed in Proposition 5.1.1.11 that if F is a left adjoint of some functor G, then it is isomorphic to every other left adjoint of G, and G is isomorphic to every other right adjoint of F.

[5] It may not be clear that by this point we have also handled the question, "is every element of U even?" One simply checks that `odd` is not an element of $\exists_p U$.

Example 5.1.2.1. Let $\mathcal{C}$ be a category and $t\colon \mathcal{C} \to \underline{1}$ the unique functor to the terminal category. Then t is a left adjoint if and only if $\mathcal{C}$ has a terminal object, and t is a right adjoint if and only if $\mathcal{C}$ has an initial object. The proofs are dual, so let's focus on the first.

The functor t has a right adjoint $R\colon \underline{1} \to \mathcal{C}$ if and only if there is an isomorphism

$$\mathrm{Hom}_{\mathcal{C}}(c, r) \cong \mathrm{Hom}_{\underline{1}}(t(c), 1),$$

where $r = R(1)$. But $\mathrm{Hom}_{\underline{1}}(t(c), 1)$ has one element. Thus t has a right adjoint iff there is a unique morphism $c \to r$ in $\mathcal{C}$. This is the definition of r being a terminal object.

When we defined colimits and limits in Definitions 4.5.3.26 and 4.5.3.19 we did so for individual I-shaped diagrams $X\colon I \to \mathcal{C}$. Using adjoints we can define the limit of every I-shaped diagram in $\mathcal{C}$ at once.

Let $t\colon \mathcal{C} \to \underline{1}$ denote the unique functor to the terminal category. Given an object $c \in \mathrm{Ob}(\mathcal{C})$, consider it as a functor $c\colon \underline{1} \to \mathcal{C}$. Then $c \circ t\colon I \to \mathcal{C}$ is the *constant functor at c*, sending each object in I to the same $\mathcal{C}$-object c, and every morphism in I to id_c. This induces a functor that we denote by $\Delta_t\colon \mathcal{C} \to \mathrm{Fun}(I, \mathcal{C})$.

Suppose we want to take the colimit or limit of X. We are given an object X of $\mathrm{Fun}(I, \mathcal{C})$ and we want back an object of $\mathcal{C}$. We could hope, and it turns out to be true, that the adjoints of Δ_t are the limit and colimit. Indeed let $\Sigma_t\colon \mathrm{Fun}(I, \mathcal{C}) \to \mathcal{C}$ be the left adjoint of Δ_t, and let $\Pi_t\colon \mathrm{Fun}(I, \mathcal{C}) \to \mathcal{C}$ be the right adjoint of Δ_t. Then Σ_t is the functor that takes colimits, and Π_t is the functor that takes limits.

We will work with a generalization of colimits and limits in Section 5.1.4. But for now, let's bring this down to earth with a concrete example.

Example 5.1.2.2. Let $\mathcal{C} = \mathbf{Set}$, and let $I = \underline{3}$. The category $\mathrm{Fun}(I, \mathbf{Set})$ is the category of $\{1, 2, 3\}$-indexed sets, e.g. $(\mathbb{Z}, \mathbb{N}, \mathbb{Z}) \in \mathrm{Ob}(\mathrm{Fun}(I, \mathbf{Set}))$ is an object of it. The functor $\Delta_t\colon \mathbf{Set} \to \mathrm{Fun}(I, \mathbf{Set})$ acts as follows. Given a set $c \in \mathrm{Ob}(\mathbf{Set})$, consider it as a functor $c\colon \underline{1} \to \mathbf{Set}$, and the composite $c \circ t\colon I \to \mathbf{Set}$ is the constant functor. That is, $\Delta_t(c)\colon I \to \mathbf{Set}$ is the $\{1, 2, 3\}$–indexed set (c, c, c).

To say that Δ_t has a right adjoint called $\Pi_t\colon \mathrm{Fun}(I, \mathbf{Set}) \to \mathbf{Set}$ and that it "takes limits" should mean that if we look through the definition of right adjoint, we will see that the formula will somehow yield the appropriate limit. Fix a functor $D\colon I \to \mathbf{Set}$, so $D(1), D(2)$, and $D(3)$ are sets. The limit $\lim D$ of D is the product $D(1) \times D(2) \times D(3)$. For example, if $D = (\mathbb{Z}, \mathbb{N}, \mathbb{Z})$ then $\lim D = \mathbb{Z} \times \mathbb{N} \times \mathbb{Z}$. How does this fact arise in the definition of adjoint?

The definition of Π_t being the right adjoint to Δ_t says that there is a natural isomorphism of sets,

$$\mathrm{Hom}_{\mathrm{Fun}(I, \mathbf{Set})}(\Delta_t(c), D) \cong \mathrm{Hom}_{\mathbf{Set}}(c, \Pi_t(D)). \tag{5.2}$$

The left-hand side has elements $f \in \mathrm{Hom}_{\mathrm{Fun}(I, \mathbf{Set})}(\Delta_t(c), D)$ that look like the left below, but having these three maps is equivalent to having the diagram to the right below:

$$\begin{tikzcd}
c \arrow[d, "f(1)"] & c \arrow[d, "f(2)"] & c \arrow[d, "f(3)"] \\
D(1) & D(2) & D(3)
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & c \arrow[dl, "f(1)"'] \arrow[d, "f(2)"] \arrow[dr, "f(3)"] & \\
D(1) & D(2) & D(3)
\end{tikzcd}$$

The isomorphism in (5.2) says that choosing the three maps $f(1), f(2), f(3)$ is the same thing as choosing a function $c \to \Pi_t(D)$. But this is very close to the universal property of limits: there is a unique map $\ell \colon c \to D(1) \times D(2) \times D(3)$, so this product serves well as Π_t as we have said. We're not giving a formal proof here, but what is missing at this point is the fact that certain diagrams have to commute. This comes down to the naturality of the isomorphism (5.2). The map ℓ induces a naturality square

$$\begin{tikzcd}
\Delta_t(c) \arrow[r, "\Delta_t(\ell)"] \arrow[d, "f"'] & \Delta_t\Pi_t D \arrow[d, "\pi"] \\
D \arrow[r, equal] & D
\end{tikzcd}$$

which says that the following diagram commutes:

$$\begin{tikzcd}
& c \arrow[dl, "f(1)"'] \arrow[d, "f(2)"] \arrow[dr, "f(3)"] \arrow[dd, bend right, "\ell"'] & \\
D(1) & D(2) & D(3) \\
& D(1) \times D(2) \times D(3) \arrow[ul, "\pi_1"] \arrow[u, "\pi_2"'] \arrow[ur, "\pi_3"'] &
\end{tikzcd}$$

It is not hard to show that the composition of left adjoints is a left adjoint, and the composition of right adjoints is a right adjoint. In the following example we show how currying (as in Sections 2.7.2 and 5.1.1.8) arises out of a certain combination of data migration functors.

Example 5.1.2.3 (Currying via Δ, Σ, Π). Let A, B, and C be sets. Consider the unique functor $a \colon A \to \underline{1}$ and consider B and C as functors $\underline{1} \xrightarrow{B} \mathbf{Set}$ and $\underline{1} \xrightarrow{C} \mathbf{Set}$ respectively.

$$\begin{tikzcd}
A \arrow[r, "a"] & \underline{1} \arrow[r, bend left, "B"] \arrow[r, bend right, "C"'] & \mathbf{Set}
\end{tikzcd}$$

Note that $\underline{1}$–$\mathbf{Set} \cong \mathbf{Set}$, and we will elide the difference. Our goal is to see currying arise out of the adjunction between $\Sigma_a \circ \Delta_a$ and $\Pi_a \circ \Delta_a$, which tells us that there is an isomorphism

$$\mathrm{Hom}_{\mathbf{Set}}(\Sigma_a \Delta_a(B), C) \cong \mathrm{Hom}_{\mathbf{Set}}(B, \Pi_a \Delta_a(C)). \tag{5.3}$$

By definition, $\Delta_a(B) \colon A \to \mathbf{Set}$ assigns to each element $a \in A$ the set B. Since Σ_A takes disjoint unions, we have a bijection

$$\Sigma_a(\Delta_a(B)) = \left(\coprod_{a \in A} B \right) \cong A \times B.$$

Similarly $\Delta_a(C)\colon A \to \mathbf{Set}$ assigns to each element $a \in A$ the set C. Since Π_A takes products, we have a bijection

$$\Pi_a(\Delta_a(C)) = \left(\prod_{a\in A} C\right) \cong C^A.$$

The currying isomorphism $\mathrm{Hom}_{\mathbf{Set}}(A \times B, C) \cong \mathrm{Hom}_{\mathbf{Set}}(B, C^A)$ falls out of (5.3).

5.1.3 Preservation of colimits or limits

One useful fact about adjunctions is that left adjoints preserve all colimits and right adjoints preserve all limits.

Proposition 5.1.3.1. *Let* $L\colon \mathcal{B} \rightleftarrows \mathcal{A} : R$ *be an adjunction. For any indexing category I and functor $D\colon I \to \mathcal{B}$, if D has a colimit in $\mathcal{B}$ then there is a unique isomorphism*

$$L(\operatorname{colim} D) \cong \operatorname{colim}(L \circ D).$$

Similarly, for any $I \in \mathrm{Ob}(\mathbf{Cat})$ and functor $D\colon I \to \mathcal{A}$, if D has a limit in $\mathcal{A}$ then there is a unique isomorphism

$$R(\lim D) \cong \lim(R \circ D).$$

Proof. The proof is simple if one knows the Yoneda lemma (Section 5.2.1.12). I have decided to skip it to keep the book shorter. See [Mac].

□

Example 5.1.3.2. Since $\mathrm{Ob}\colon \mathbf{Cat} \to \mathbf{Set}$ is both a left adjoint and a right adjoint, it must preserve both limits and colimits. This means that if you want to know the set of objects in the fiber product of some categories, you can simply take the fiber product of the set of objects in those categories,

$$\mathrm{Ob}(\mathcal{A} \times_{\mathcal{C}} \mathcal{B}) \cong \mathrm{Ob}(\mathcal{A}) \times_{\mathrm{Ob}(\mathcal{C})} \mathrm{Ob}(\mathcal{B}).$$

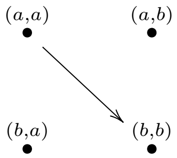
While the right-hand side might look daunting, it is just a fiber product in **Set** which is quite understandable.

This is greatly simplifying. If one thinks through what defines a limit in **Cat**, one is dragged through notions of slice categories and terminal objects in them. These slice categories are in **Cat** so they involve several categories and functors, and it gets hairy or even hopeless to a beginner. Knowing that the objects are given by a simple fiber product makes the search for limits in **Cat** much simpler.

For example, if $[n]$ is the linear order category of length n then $[n] \times [m]$ has $nm + n + m + 1$ objects because $[n]$ has $n + 1$ objects and $[m]$ has $m + 1$ objects.

Example 5.1.3.3. The "path poset" functor $L\colon \mathbf{Grph} \to \mathbf{PrO}$ given by existence of paths (see Exercise 4.1.2.11) is left adjoint to the functor $R\colon \mathbf{PrO} \to \mathbf{Grph}$ given by replacing $\leqslant$'s by arrows. This means that L preserves colimits. So taking the union of graphs G and H results in a graph whose path poset $L(G \sqcup H)$ is the union of the path posets of G and H. But this is not so for products.

Let $G = H = \boxed{\overset{a}{\bullet}\xrightarrow{f}\overset{b}{\bullet}}$. Then $L(G) = L(H) = [1]$, the linear order of length 1. But the product $G \times H$ in **Grph** looks like the graph



Its preorder $L(G \times H)$ does not have $(a,a) \leqslant (a,b)$, whereas this is the case in $L(G) \times L(H)$.

5.1.4 Data migration

As we saw in Sections 4.2.2 and 4.2.2.5, a database schema is a category $\mathcal{C}$ and an instance is a functor $I\colon \mathcal{C} \to \mathbf{Set}$.

Notation 5.1.4.1. Let $\mathcal{C}$ be a category. Throughout this section we denote by $\mathcal{C}$–**Set** the category $\mathrm{Fun}(\mathcal{C}, \mathbf{Set})$ of functors from $\mathcal{C}$ to **Set**, i.e. the category of instances on $\mathcal{C}$.

In this section we discuss what happens to the resulting instances when different schemas are connected by a functor, say $F\colon \mathcal{C} \to \mathcal{D}$. It turns out that three adjoint functors emerge: $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$, $\Sigma_F\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$, and $\Pi_F\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$, where Δ_F is adjoint to both,

$$\Sigma_F\colon \mathcal{C}\text{–}\mathbf{Set} \rightleftarrows \mathcal{D}\text{–}\mathbf{Set} :\!\Delta_F \qquad\qquad \Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \rightleftarrows \mathcal{C}\text{–}\mathbf{Set} :\!\Pi_F.$$

It turns out that almost all the basic database operations are captured by these three functors. For example, Δ_F handles the job of duplicating or deleting tables, as well as duplicating or deleting columns in a single table. The functor Σ_F handles taking unions, and the functor Π_F handles joining tables together, matching columns, or selecting the rows with certain properties (e.g. everyone whose first name is Mary).

5.1.4.2 Pullback: Δ

Given a functor $F\colon \mathcal{C} \to \mathcal{D}$ and a functor $I\colon \mathcal{D} \to \mathbf{Set}$, we can compose them to get a functor $I \circ F\colon \mathcal{C} \to \mathbf{Set}$. In other words, the presence of F provides a way to convert $\mathcal{D}$-instances into $\mathcal{C}$-instances. In fact this conversion is functorial, meaning that morphisms of $\mathcal{D}$-instances are sent to morphisms of $\mathcal{C}$-instances. We denote the resulting functor by $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$ and call it *pullback along F*.

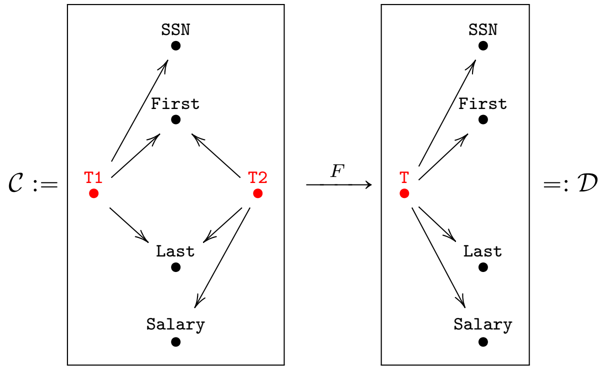
We have seen an example of this before in Example 4.3.2.15, where we showed how a monoid homomorphism $F\colon \mathcal{M}' \to \mathcal{M}$ could add functionality to a finite state machine. More generally, we can use pullbacks to reorganize data, copying and deleting tables and columns.

Remark 5.1.4.3. Given a functor $F\colon \mathcal{C} \to \mathcal{D}$, which we think of as a schema translation, the functor $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$ "goes the opposite way". The reasoning is simple to any explain (composition of functors) but something about it is often very strange to people, at first. The rough idea of this "contravariance" is captured by the role-reversal in the following slogan:

Slogan 5.1.4.4.

> “ *If I get my information from you, then your information becomes my information.* ”

Consider the following functor $F\colon \mathcal{C} \to \mathcal{D}$: [6]



(5.4)

Let’s spend a moment recalling how to “read” schemas. In schema $\mathcal{C}$ there are leaf tables `SSN, First, Last, Salary`, which represent different kinds of basic data. More interestingly, there are two *fact tables*. The first is called `T1` and it relates `SSN, First,` and `Last`. The second is called `T2` and it relates `First, Last,` and `Salary`.

The functor $F\colon \mathcal{C} \to \mathcal{D}$ relates $\mathcal{C}$ to a schema with a single fact table relating all four attributes: `SSN, First, Last,` and `Salary`. We are interested in $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$. Suppose given the following database instance $I\colon \mathcal{D} \to \mathbf{Set}$ on $\mathcal{D}$:

T				
ID	**SSN**	**First**	**Last**	**Salary**
XF667	115-234	Bob	Smith	$250
XF891	122-988	Sue	Smith	$300
XF221	198-877	Alice	Jones	$100

SSN
ID
115-234
118-334
122-988
198-877
342-164

First
ID
Adam
Alice
Bob
Carl
Sam
Sue

Last
ID
Jones
Miller
Pratt
Richards
Smith

Salary
ID
$100
$150
$200
$250
$300

How do you get the instance $\Delta_F(I)\colon \mathcal{C} \to \mathbf{Set}$? The formula was given above: compose I with F. In terms of tables, it feels like duplicating table `T` as `T1` and `T2`, but deleting a column from each in accordance with the definition of $\mathcal{C}$ in (5.4). Here is the result, $\Delta_F(I)$, in table form:

[6] This example was taken from [Sp1], `http://arxiv.org/abs/1009.1166`.

T1			
ID	**SSN**	**First**	**Last**
XF667	115-234	Bob	Smith
XF891	122-988	Sue	Smith
XF221	198-877	Alice	Jones

T2			
ID	**First**	**Last**	**Salary**
XF221	Alice	Jones	$100
XF667	Bob	Smith	$250
XF891	Sue	Smith	$300

SSN
ID
115-234
118-334
122-988
198-877
342-164

First
ID
Adam
Alice
Bob
Carl
Sam
Sue

Last
ID
Jones
Miller
Pratt
Richards
Smith

Salary
ID
$100
$150
$200
$250
$300

Exercise 5.1.4.5. Let $\mathcal{C} = (G, \simeq)$ be a schema. A leaf table is an object $c \in \mathrm{Ob}(\mathcal{C})$ with no outgoing arrows.

a.) Write the condition of being a "leaf table" mathematically in three different languages: that of graphs (using symbols V, A, src, tgt), that of categories (using $\mathrm{Hom}_{\mathcal{C}}$, etc.), and that of tables (in terms of columns, tables, rows, etc.).

b.) In the language of categories, is there a difference between a terminal object and a leaf table? Explain.

◊

Exercise 5.1.4.6. Consider the schemas

$$[1] = \boxed{\overset{0}{\bullet} \xrightarrow{f} \overset{1}{\bullet}} \qquad \text{and} \qquad [2] = \boxed{\overset{0}{\bullet} \xrightarrow{g} \overset{1}{\bullet} \xrightarrow{h} \overset{2}{\bullet}},$$

and the functor $F\colon [1] \to [2]$ given by sending $0 \mapsto 0$ and $1 \mapsto 2$.

a.) How many possibilities are there for $F(f)$?

b.) Now suppose $I\colon [2] \to \mathbf{Set}$ is given by the following tables.

0	
ID	**g**
Am	To be verb
Baltimore	Place
Carla	Person
Develop	Action verb
Edward	Person
Foolish	Adjective
Green	Adjective

1	
ID	**h**
Action verb	Verb
Adjective	Adjective
Place	Noun
Person	Noun
To be verb	Verb

2
ID
Adjective
Noun
Verb

Write out the two tables associated to the [1]-instance $\Delta_F(I)\colon [1] \to \mathbf{Set}$.

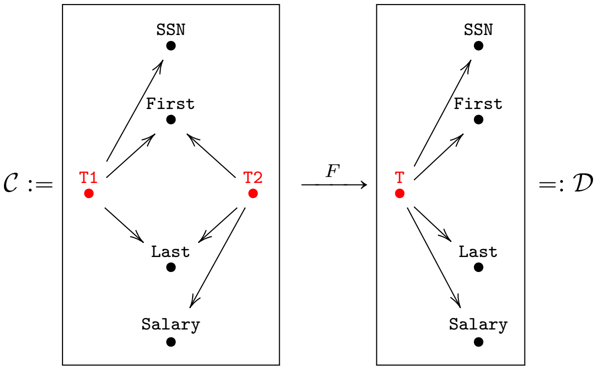
◊

5.1.4.7 Left pushforward: Σ

Let $F\colon \mathcal{C} \to \mathcal{D}$ be a functor. The functor $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$ has a left adjoint, $\Sigma_F\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$. The rough idea is that Σ_F performs parameterized colimits. Given an instance $I\colon \mathcal{C} \to \mathbf{Set}$, we get an instance on $\mathcal{D}$ that acts as follows. For each object $d \in \mathrm{Ob}(\mathcal{D})$, the set $\Sigma_F(I)(d)$ is the colimit (think, union) of some diagram back home in $\mathcal{C}$.

Left pushforwards (also known as left Kan extensions) are discussed at length in [Sp1]; here we begin with some examples from that paper.

Example 5.1.4.8. We again use the functor $F\colon \mathcal{C} \to \mathcal{D}$ drawn below



(5.4)

We will be applying the left pushforward $\Sigma_F\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$ to the following instance $I\colon \mathcal{C} \to \mathbf{Set}$:

T1

ID	SSN	First	Last
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

T2

ID	First	Last	Salary
T2-001	Alice	Jones	\$100
T2-002	Sam	Miller	\$150
T2-004	Sue	Smith	\$300
T2-010	Carl	Pratt	\$200

SSN

ID
115-234
118-334
122-988
198-877
342-164

First

ID
Adam
Alice
Bob
Carl
Sam
Sue

Last

ID
Jones
Miller
Pratt
Richards
Smith

Salary

ID
\$100
\$150
\$200
\$250
\$300

The functor $F\colon \mathcal{C} \to \mathcal{D}$ sent both tables `T1` and `T2` to table `T`. Applying Σ_F will take the what was in `T1` and `T2` and put the union in `T`. The result $\Sigma_F I\colon \mathcal{D} \to \mathbf{Set}$ is as follows:

T				
ID	**SSN**	**First**	**Last**	**Salary**
T1-001	115-234	Bob	Smith	T1-001.Salary
T1-002	122-988	Sue	Smith	T1-002.Salary
T1-003	198-877	Alice	Jones	T1-003.Salary
T2-001	T2-A101.SSN	Alice	Jones	$100
T2-002	T2-A102.SSN	Sam	Miller	$150
T2-004	T2-004.SSN	Sue	Smith	$300
T2-010	T2-A110.SSN	Carl	Pratt	$200

SSN
ID
115-234
118-334
122-988
198-877
342-164
T2-001.SSN
T2-002.SSN
T2-004.SSN
T2-010.SSN

First
ID
Adam
Alice
Bob
Carl
Sam
Sue

Last
ID
Jones
Miller
Pratt
Richards
Smith

Salary
ID
$100
$150
$200
$250
$300
T1-001.Salary
T1-002-Salary
T1-003-Salary

As you can see, there was no set salary information for any data coming from table `T1` nor any set SSN information for any data coming form table `T2`. But the definition of adjoint, given in Definition 5.1.1.1, yielded the universal response: freely add new variables that take the place of missing information. It turns out that this idea already has a name in logic, *Skolem variables*, and a name in database theory, *labeled nulls*.

Exercise 5.1.4.9. Consider the functor $F\colon \underline{3} \to \underline{2}$ sending $1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 2$.

a.) Write down an instance $I\colon \underline{3} \to \textbf{Set}$.

b.) Given the description that "Σ_F performs a parameterized colimit", make an educated guess about what $\Sigma_F(I)$ will be. Give your answer in the form of two sets that are made up from the three sets you already wrote down.

◊

We now briefly give the actual formula for computing left pushforwards. Suppose that $F\colon \mathcal{C} \to \mathcal{D}$ is a functor and let $I\colon \mathcal{C} \to \textbf{Set}$ be a set-valued functor on $\mathcal{C}$. Then $\Sigma_F(I)\colon \mathcal{D} \to \textbf{Set}$ is defined as follows. Given an object $d \in \text{Ob}(\mathcal{D})$ we first form the comma category (see Definition 4.6.4.1) for the setup

$$\mathcal{C} \xrightarrow{F} \mathcal{D} \xleftarrow{d} \underline{1}$$

and denote it by $(F \downarrow d)$. There is a canonical projection functor $\pi\colon (F \downarrow d) \to \mathcal{C}$, which we can compose with $I\colon \mathcal{C} \to \textbf{Set}$ to obtain a functor $(F \downarrow d) \to \textbf{Set}$. We are ready to define $\Sigma_F(I)(d)$ to be its colimit,

$$\Sigma_F(I)(d) := \underset{(F\downarrow d)}{\text{colim}}\, I \circ \pi.$$

We have defined $\Sigma_F(I)\colon \mathcal{D} \to \textbf{Set}$ on objects $d \in \text{Ob}(\mathcal{D})$. As for morphisms we will be even more brief, but one can see [Sp1] for details. Given a morphism $g\colon d \to d'$ one notes that there is an induced functor $(F \downarrow g) \colon (F \downarrow d) \to (F \downarrow d')$ and a commutative diagram of categories:

$$\begin{tikzcd}
(F\downarrow d) \arrow[rr, "(F\downarrow g)"] \arrow[dr, "\pi"] \arrow[ddr, "I\circ\pi"'] & & (F\downarrow d') \arrow[dl, "\pi'"'] \arrow[ddl, "I\circ\pi'"] \\
& \mathcal{C} \arrow[d, "I"] & \\
& \mathbf{Set} &
\end{tikzcd}$$

By the universal property of colimits, this induces the required function

$$\operatorname*{colim}_{(F\downarrow d)} I \circ \pi \xrightarrow{\Sigma_F(I)(g)} \operatorname*{colim}_{(F\downarrow d')} I \circ \pi'.$$

5.1.4.10 Right pushforward: Π

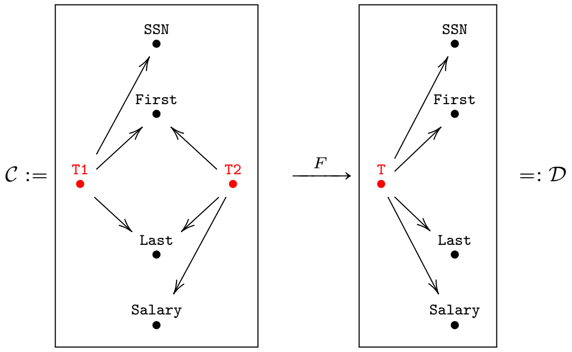
Let $F \colon \mathcal{C} \to \mathcal{D}$ be a functor. We heard in Section 5.1.4.7 that the functor $\Delta_F \colon \mathcal{D}$–$\mathbf{Set} \to \mathcal{C}$–$\mathbf{Set}$ has a left adjoint. Here we explain that it has a right adjoint, $\Pi_F \colon \mathcal{C}$–$\mathbf{Set} \to \mathcal{D}$–$\mathbf{Set}$ as well. The rough idea is that Π_F performs parameterized limits. Given an instance $I \colon \mathcal{C} \to \mathbf{Set}$, we get an instance on $\mathcal{D}$ that acts as follows. For each object $d \in \mathrm{Ob}(\mathcal{D})$, the set $\Pi_F(I)(d)$ is the limit (think, fiber product) of some diagram back home in $\mathcal{C}$.

Right pushforwards (also known as right Kan extensions) are discussed at length in [Sp1]; here we begin with some examples from that paper.

Example 5.1.4.11. We once again use the functor $F \colon \mathcal{C} \to \mathcal{D}$ from Example 5.1.4.8. We will apply the right pushforward Π_F to instance $I \colon \mathcal{C} \to \mathbf{Set}$ from that example. [7]

The instance $\Pi_F(I)$ will put data in all 5 tables in $\mathcal{D}$. In T it will put pairs (t_1, t_2) where t_1 is a row in T1 and t_2 is a row in T2 for which the first and last names agree.

[7] To repeat for convenience,



(5.4)

$I \colon \mathcal{C} \to \mathbf{Set}$ is

T1

ID	SSN	First	Last
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

T2

ID	First	Last	Salary
T2-001	Alice	Jones	\$100
T2-002	Sam	Miller	\$150
T2-004	Sue	Smith	\$300
T2-010	Carl	Pratt	\$200

It will copy the leaf tables exactly, so we do not display them here; the following is the table T for $\Pi_F(I)$:

T				
ID	**SSN**	**First**	**Last**	**Salary**
T1-002T2-A104	122-988	Sue	Smith	\$300
T1-003T2-A101	198-877	Alice	Jones	\$100

Looking at T1 and T2, there were only two ways to match first and last names.

Exercise 5.1.4.12. Consider the functor $F\colon \underline{3} \to \underline{2}$ sending $1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 2$.

a.) Write down an instance $I\colon \underline{3} \to \mathbf{Set}$.

b.) Given the description that "Π_F performs a parameterized limit", make an educated guess about what $\Pi_F(I)$ will be. Give your answer in the form of two sets that are made up from the three sets you already wrote down.

◊

We now briefly give the actual formula for computing right pushforwards. Suppose that $F\colon \mathcal{C} \to \mathcal{D}$ is a functor and let $I\colon \mathcal{C} \to \mathbf{Set}$ be a set-valued functor on $\mathcal{C}$. Then $\Pi_F(I)\colon \mathcal{D} \to \mathbf{Set}$ is defined as follows. Given an object $d \in \mathrm{Ob}(\mathcal{D})$ we first form the comma category (see Definition 4.6.4.1) for the setup

$$\underline{1} \xrightarrow{d} \mathcal{D} \xleftarrow{F} \mathcal{C}$$

and denote it by $(d \downarrow F)$. There is a canonical projection functor $\pi\colon (d \downarrow F) \to \mathcal{C}$, which we can compose with $I\colon \mathcal{C} \to \mathbf{Set}$ to obtain a functor $(d \downarrow F) \to \mathbf{Set}$. We are ready to define $\Pi_F(I)(d)$ to be its limit,

$$\Pi_F(I)(d) := \lim_{(d \downarrow F)} I \circ \pi.$$

We have defined $\Pi_F(I)\colon \mathcal{D} \to \mathbf{Set}$ on objects $d \in \mathrm{Ob}(\mathcal{D})$. As for morphisms we will be even more brief, but one can see [Sp1] for details. Given a morphism $g\colon d \to d'$ one notes that there is an induced functor $(g \downarrow F)\colon (d' \downarrow F) \to (d \downarrow F)$ and a commutative diagram of categories:

$$\begin{tikzcd}
(d' \downarrow F) \arrow[rr, "(g\downarrow F)"] \arrow[dr, "\pi'"] \arrow[ddr, "I\circ\pi'"'] & & (d \downarrow F) \arrow[dl, "\pi"'] \arrow[ddl, "I\circ\pi"] \\
& \mathcal{C} \arrow[d, "I"] & \\
& \mathbf{Set} &
\end{tikzcd}$$

SSN
ID
115-234
118-334
122-988
198-877
342-164

First
ID
Adam
Alice
Bob
Carl
Sam
Sue

Last
ID
Jones
Miller
Pratt
Richards
Smith

Salary
ID
\$100
\$150
\$200
\$250
\$300

By the universal property of limits, this induces the required function

$$\lim_{(d\downarrow F)} I \circ \pi \xrightarrow{\Pi_F(I)(g)} \lim_{(d'\downarrow F)} I \circ \pi'.$$

5.2 Categories of functors

For any two categories $\mathcal{C}$ and $\mathcal{D}$, [8] we discussed the category $\mathrm{Fun}(\mathcal{C}, \mathcal{D})$ of functors and natural transformations between them. In this section we discuss functor categories a bit more and give some important applications within mathematics (sheaves) that extend to the real world.

5.2.1 Set-valued functors

Let $\mathcal{C}$ be a category. Then we have been writing $\mathcal{C}$–**Set** to denote the functor category $\mathrm{Fun}(\mathcal{C}, \mathbf{Set})$. Here is a nice result about these categories.

Proposition 5.2.1.1. *Let $\mathcal{C}$ be a category. The category $\mathcal{C}$–**Set** is closed under colimits and limits.*

Sketch of proof. Let J be an indexing category and $D\colon J \to \mathcal{C}$–**Set** a functor. For each object $c \in \mathrm{Ob}(\mathcal{C})$, we have a functor $D_c\colon J \to \mathbf{Set}$ defined by $D_c(j) = D(j)(c)$. Define a functor $L\colon \mathcal{C} \to \mathbf{Set}$ by $L(c) = \lim_J D_c$, and note that for each $f\colon c \to c'$ in $\mathcal{C}$ there is an induced function $L(f)\colon L(c) \to L(c')$. One can check that L is a limit of J, because it satisfies the relevant universal property.

The dual proof holds for colimits.

□

Application 5.2.1.2. When taking in data about a scientific subject, one often finds that the way one thinks about the problem changes over time. We understand this phenomenon in the language of databases in terms of a series of schemas $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$, perhaps indexed chronologically. The problem is that old data is held in old schemas and we want to see it in our current understanding. The first step is to transfer all the old data to our new schema in the freest possible way, that is, making no assumptions about how to fill in the new fields. If one creates functors $F_i\colon \mathcal{C}_i \to \mathcal{C}_{i+1}$ from each of these schemas to the next, then we can push the data forward using Σ_{F_i}.

Doing this we will have n datasets on $\mathcal{D} := \mathcal{C}_n$, namely one for each "epoch of understanding". Since the category $\mathcal{D}$–**Set** has all colimits, we can take the union of these datasets and get one. It will have many Skolem variables (see Example 5.1.4.8), and these need to be handled in a coherent way. However, the universality of left adjoints could be interpreted as saying that any reasonable formula for handling this old data can be applied to our results.

◊◊

Exercise 5.2.1.3. By Proposition 5.2.1.1, the category $\mathcal{C}$–**Set** is closed under taking limits. By Exercises 4.5.3.21 and 4.5.3.28, this means in particular that $\mathcal{C}$–**Set** has an initial object and a terminal object.

a.) Let $A \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$ be the initial object, considered as a functor $A\colon \mathcal{C} \to \mathbf{Set}$. For any $c \in \mathrm{Ob}(\mathcal{C})$, what is the set $A(c)$?

[8] Technically $\mathcal{C}$ has to be small (see Remark 4.1.1.2), but as we said there, we are not worrying about that distinction in this book.

b.) Let $Z \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$ be the terminal object, considered as a functor $Z\colon \mathcal{C} \to \mathbf{Set}$. For any $c \in \mathrm{Ob}(\mathcal{C})$, what is the set $Z(c)$?

$\Diamond$

Proposition 5.2.1.1 says that we can add or multiply database states together. In fact, database states on $\mathcal{C}$ form what is called a *topos* which means that just about every consideration we made for sets holds for instances on any schema. Perhaps the simplest schema is $\mathcal{C} = \boxed{\bullet}$, on which the relevant topos is indeed **Set**. But schemas can be arbitrarily complex, and it is impressive that all of these considerations make sense in such generality. Here is a table that makes a comparison between these domains.

Dictionary between **Set** and $\mathcal{C}$–**Set**	
Concept in Set	**Concept in $\mathcal{C}$–Set**
Set	Object in $\mathcal{C}$–**Set**
Function	Morphism in $\mathcal{C}$–**Set**
Element	Representable functor
Empty set	Initial object
Natural numbers	Natural numbers object
Image	Image
(Co)limits	(Co)limits
Exponential objects	Exponential objects
“Familiar” arithmetic	“Familiar” arithmetic
Power sets 2^X	Power objects Ω^X
Characteristic functions	Characteristic morphisms
Surjections, injections	Epimorphisms, monomorphisms

In the above table we said that elements of a set are akin to representable functors in $\mathcal{C}$–**Set**, but we have not yet defined those; we do so in Section 5.2.1.6. First we briefly discuss monomorphisms and epimorphisms in general (Definition 5.2.1.4) and then in $\mathcal{C}$–**Set** (Proposition 5.2.1.5).

Definition 5.2.1.4 (Monomorphism, Epimorphism)**.** Let $\mathcal{S}$ be a category and let $f\colon X \to Y$ be a morphism. We say that f is a *monomorphism* if it has the following property. For all objects $A \in \mathrm{Ob}(\mathcal{S})$ and morphisms $g, g'\colon A \to X$ in $\mathcal{S}$,

$$\begin{tikzcd}
A \arrow[r, bend left, "g"] \arrow[r, bend right, "g'"'] & X \arrow[r, "f"] & Y
\end{tikzcd}$$

if $f \circ g = f \circ g'$ then $g = g'$.

We say that $f\colon X \to Y$ is an *epimorphism* if it has the following property. For all objects $B \in \mathrm{Ob}(\mathcal{S})$ and morphisms $h, h'\colon Y \to B$ in $\mathcal{S}$,

$$\begin{tikzcd}
X \arrow[r, "f"] & Y \arrow[r, bend left, "h"] \arrow[r, bend right, "h'"'] & B
\end{tikzcd}$$

if $h \circ f = h' \circ f$ then $h = h'$.

In the category of sets, monomorphisms are the same as injections and epimorphisms are the same as surjections (see Proposition 2.7.5.4). The same is true in $\mathcal{C}$–**Set**: one can check "table by table" that a morphism of instances is mono or epi.

Proposition 5.2.1.5. *Let $\mathcal{C}$ be a category and let $X, Y : \mathcal{C} \to \mathbf{Set}$ be objects in $\mathcal{C}$–**Set** and let $f \colon X \to Y$ be a morphism in $\mathcal{C}$–**Set**. Then f is a monomorphism (respectively an epimorphism) if and only if, for every object $c \in \mathrm{Ob}(\mathcal{C})$, the function $f(c) \colon X(c) \to Y(c)$ is injective (respectively surjective).*

Sketch of proof. We first show that if f is mono (respectively epi) then so is $f(c)$ for all $c \in \mathrm{Ob}(\mathcal{C})$. Considering c as a functor $c \colon \underline{1} \to \mathcal{C}$, this result follows from the fact that Δ_c preserves limits and colimits, hence monos and epis.

We now check that if $f(c)$ is mono for all $c \in \mathrm{Ob}(\mathcal{C})$ then f is mono. Suppose that $g, g' \colon A \to X$ are morphisms in $\mathcal{C}$–**Set** such that $f \circ g = f \circ g'$. Then for every c we have $f \circ g(c) = f \circ g'(c)$ which implies by hypothesis that $g(c) = g'(c)$. But the morphisms in $\mathcal{C}$–**Set** are natural transformations, and if two natural transformations g, g' have the same components then they are the same.

A similar argument works to show the analogous result for epimorphisms.

□

5.2.1.6 Representable functors

Given a category $\mathcal{C}$, there are certain functors $\mathcal{C} \to \mathbf{Set}$ that come with the package, one for every object in $\mathcal{C}$. So if $\mathcal{C}$ is a database schema, then for every table $c \in \mathrm{Ob}(\mathcal{C})$ there is a certain database instance associated to it. These instances, i.e. set-valued functors, are called representable functors, and they'll be defined in Definition **??**. The idea is that if a database schema represents a conceptual layout of types (e.g. as an olog), then each type T has an instance associated to it, standing for "the generic thing of type T with all its generic attributes".

Definition 5.2.1.7. Let $\mathcal{C}$ be a category and let $c \in \mathrm{Ob}(\mathcal{C})$ be an object. The functor $\mathrm{Hom}_{\mathcal{C}}(c, -) \colon \mathcal{C} \to \mathbf{Set}$, sending $d \in \mathrm{Ob}(\mathcal{C})$ to the set $\mathrm{Hom}_{\mathcal{C}}(c, d)$ and acting similarly on morphisms $d \to d'$, is said to be *represented by c*. If a functor $F \colon \mathcal{C} \to \mathbf{Set}$ is isomorphic to $\mathrm{Hom}_{\mathcal{C}}(c, -)$, we say that F is *a representable functor*. We sometimes write $Y_c := \mathrm{Hom}_{\mathcal{C}}(c, -)$ for short.

Example 5.2.1.8. Given a category $\mathcal{C}$ and an object $c \in \mathrm{Ob}(\mathcal{C})$, we get a representable functor. If we think of $\mathcal{C}$ as a database schema and c as a table, then what does the representable functor $Y_c \colon \mathcal{C} \to \mathbf{Set}$ look like in terms of databases? It turns out that the following procedure will generate it.

Begin by writing a new row, say "☺", in the ID column of table c. For each foreign key column $f \colon c \to c'$, add a row in the ID column of table c' called "f(☺)" and record that result (i.e. "f(☺)") in the f column of table c. Repeat as follows: for each table d, identify all rows r that have blank cell in column $g \colon d \to e$. Add a new row called "$g(r)$" to table e and record that result in the (r, g) cell of table d.

Here is a concrete example. Let $\mathcal{C}$ be the following schema:

$$\begin{tikzcd}
\overset{A}{\bullet} \arrow[r, "f"] & \overset{B}{\bullet} \arrow[r, shift left, "g_1"] \arrow[r, shift right, "g_2"'] \arrow[d, "h"'] & \overset{C}{\bullet} \arrow[r, "i"] & \overset{D}{\bullet} \\
 & \overset{E}{\bullet} & &
\end{tikzcd}$$

Then $Y_B : \mathcal{C} \to \mathbf{Set}$ is the following instance

A	
ID	f

B			
ID	g_1	g_2	h
☺	$g_1(☺)$	$g_2(☺)$	$h(☺)$

C	
ID	i
$g_1(☺)$	$i(g_1(☺))$
$g_2(☺)$	$i(g_2(☺))$

D
ID
$i(g_1(☺))$
$i(g_2(☺))$

E
ID
$h(☺)$

We began with a single element in table B and followed the arrows, putting new entries wherever they were required. One might call this the *schematically implied reference spread* or *SIRS* of the element ☺ in table B. Notice that the table at A is empty, because there are no morphisms $B \to A$.

Representable functors Y_c yield databases states that are as free as possible, subject to having the initial row ☺ in table c. We have seen things like this before (by the name of Skolem variables) when studying the left pushforward Σ. Indeed, if $c \in \mathrm{Ob}(\mathcal{C})$ is an object, we can consider it as a functor $c : \underline{1} \to \mathcal{C}$. A database instance on $\underline{1}$ is the same thing as a set X. The left pushforward $\Sigma_c(X)$ has the same kinds of Skolem variables. If $X = \{☺\}$ is a one element set, then we get the representable functor $\Sigma_c(\{☺\}) \cong Y_c$.

Exercise 5.2.1.9. Consider the schema for graphs,

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

a.) Write down the representable functor $Y_{Ar} : \mathbf{GrIn} \to \mathbf{Set}$ as two tables.

b.) Write down the representable functor Y_{Ve} as two tables.

◊

Exercise 5.2.1.10. Consider the loop schema

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

What is the representable functor $Y_s : \mathcal{L}oop \to \mathbf{Set}$? ◊

Let B be a box in an olog, say ⌜a person⌝, and recall that an aspect of B is an outgoing arrow, such as ⌜a person⌝ $\xrightarrow{\text{has as height in inches}}$ ⌜an integer⌝. The following slogan explains representable functors in those terms.

Slogan 5.2.1.11.

> " *The functor represented by* ⌜a person⌝ *simply leaves a placeholder, like* ⟨*person's name here*⟩ *or* ⟨*person's height here*⟩, *for every aspect of* ⌜a person⌝.
>
> *In general, there is a representable functor for every type in an olog. The representable functor for type T simply encapsulates the most generic or abstract example of type T, by leaving a placeholder for each of its attributes.*
> "

5.2.1.12 Yoneda's lemma

One of the most powerful tools in category theory is Yoneda's lemma. It is often considered by new students to be quite abstract, but grounding it in databases may help.

The idea is this. Suppose that $I\colon \mathcal{C} \to \textbf{Set}$ is a database instance, and let $c \in \text{Ob}(\mathcal{C})$ be an object. Because I is a functor, we know that for every row $r \in I(c)$ in table c a value has been recorded in the f-column, where $f\colon c \to c'$ is any outgoing arrow. The value in the (r, f)-cell refers to some row in table c'. What we're saying is that each row in table c induces SIRS throughout the database. They may not be "Skolem", or in any sense "freely generated", but they are there nonetheless. The point is that to each row in c there is a unique mapping $Y_c \to I$.

Lemma 5.2.1.13 (Yoneda's lemma, part 1.)**.** *Let $\mathcal{C}$ be a category, $c \in \text{Ob}(\mathcal{C})$ an object, and $I\colon \mathcal{C} \to \textbf{Set}$ a set-valued functor. There is a natural bijection*

$$\text{Hom}_{\mathcal{C}\text{-}\textbf{Set}}(Y_c, I) \xrightarrow{\cong} I(c).$$

Proof. See [Mac].

□

Example 5.2.1.14. Consider the category $\mathcal{C}$ drawn below:

$$\text{mother} \circ \text{firstChild} = \text{id}_{\text{Mother}}$$

$$\begin{tikzcd} \mathcal{C} := & \overset{\texttt{Child}}{\bullet} \arrow[r, "\text{mother}"] & \overset{\texttt{Mother}}{\bullet} \arrow[l, bend left, "\text{firstChild}"] \end{tikzcd}$$

There are two representable functors, $Y_{\texttt{Child}}$ and $Y_{\texttt{Mother}}$. The latter, when written as a database instance, will consist of a single row in each table. The former, $Y_{\texttt{Child}}\colon \mathcal{C} \to \textbf{Set}$ is shown here:

Child	
ID	**mother**
☺	mother(☺)
firstChild(mother(☺))	mother(☺)

Mother	
ID	**firstChild**
mother(☺)	firstChild(mother(☺))

The representable functor $Y_{\texttt{Child}}$ is the freest instance possible, starting with one element in the Child table and satisfying the constraints.

Here is another instance $I\colon \mathcal{C} \to \textbf{Set}$:

Child	
ID	**mother**
Amy	Ms. Adams
Bob	Ms. Adams
Carl	Ms. Jones
Deb	Ms. Smith

Mother	
ID	**firstChild**
Ms. Adams	Bob
Ms. Jones	Carl
Ms. Smith	Deb

Yoneda's lemma (5.2.1.13) is about the set of natural transformations $Y_{\texttt{Child}} \to I$. Recall from Definition 4.3.1.2 that a search for natural transformations can get a bit tedious. Yoneda's lemma makes the calculation quite trivial. In our case there are exactly four such natural transformations, and they are completely determined by where ☺ goes. In some sense the symbol ☺ *represents* child-ness in our database.

Exercise 5.2.1.15. Consider the schema $\mathcal{C}$ and instance $I\colon \mathcal{C} \to \mathbf{Set}$ from Example 5.2.1.14. Let $Y_{\texttt{Child}}$ be the representable functor as above.

a.) Let $\alpha\colon Y_{\texttt{Child}} \to I$ be the natural transformation sending ☺ to Amy. What is α_{Child}(firstChild(mother(☺)))? [9]

b.) Let $\alpha\colon Y_{\texttt{Child}} \to I$ be the natural transformation sending ☺ to Bob. What is $\alpha_{\texttt{Child}}$(firstChild(mother(☺)))?

c.) Let $\alpha\colon Y_{\texttt{Child}} \to I$ be the natural transformation sending ☺ to Carl. What is $\alpha_{\texttt{Child}}$(firstChild(mother(☺)))?

d.) Let $\alpha\colon Y_{\texttt{Child}} \to I$ be the natural transformation sending ☺ to Deb. What is $\alpha_{\texttt{Child}}$(firstChild(mother(☺)))?

e.) Let $\alpha\colon Y_{\texttt{Child}} \to I$ be the natural transformation sending ☺ to Amy. What is $\alpha_{\texttt{Mother}}$(mother(☺))?

◊

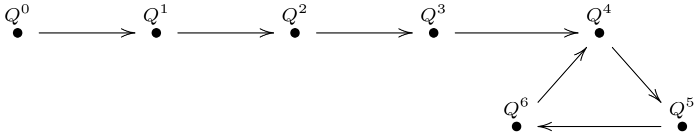
We saw in Section 5.2.1.6 that a representable functor is a mathematically-generated database instance for an abstract thing of type T. It creates placeholders for every attribute that things of type T are supposed to have.

Slogan 5.2.1.16.

> “ *Yoneda's lemma says the following. Specifying an actual thing of type T is the same as filling in all placeholders found in the generic thing of type T.* ”

Yoneda's lemma is considered by many category theory lovers to be the most important tool in the subject. While its power is probably unclear to students whose sole background in category theory comes from this book, Yoneda's lemma is indeed extremely useful for reasoning. It allows us to move the notion of functor application into the realm of morphisms between functors (i.e. morphisms in $\mathcal{C}$–**Set**, which are natural transformations). This keeps everything in one place — it's all in the morphisms — and thus more interoperable.

Example 5.2.1.17. In Example 3.1.1.26, we discussed the cyclic monoid $\mathcal{M}$ generated by the symbol Q and subject to the relation $Q^7 = Q^4$. We drew a picture like this:



(5.5)

We are finally ready to give the mathematical foundation for this picture. Since $\mathcal{M}$ is a category with one object, ▲, there is a unique representable functor (up to isomorphism) $Y := Y_{\blacktriangle}\colon \mathcal{M} \to \mathbf{Set}$. A functor $\mathcal{M} \to \mathbf{Set}$ can be thought of as a set with an $\mathcal{M}$-action, as discussed in Section 4.2.1.1. Here the required set is

$$Y(\blacktriangle) = \mathrm{Hom}_{\mathcal{M}}(\blacktriangle, \blacktriangle) \cong \{Q^0, Q^1, Q^2, Q^3, Q^4, Q^5, Q^6\}$$

[9] There is a lot of clutter, perhaps. Note that “firstChild(mother(☺))” is a row in the `Child` table. Assuming that the math follows the meaning, if ☺ points to Amy, where should firstChild(Mother(☺)) point?

and the action is pretty straightforward (it is called the *principal action*). We might say that (5.5) is a picture of this principal action of $\mathcal{M}$.

However, we can go one step further. Given a functor $Y\colon \mathcal{M} \to \mathbf{Set}$, we can take its category of elements, $\int_{\mathcal{M}} Y$ as in Section 4.6.2. The category $\int_{\mathcal{M}} Y$ has objects $Y(\blacktriangle) \in \mathrm{Ob}(\mathbf{Set})$, i.e. the set of dots in (5.5), and it has a unique morphism $Q^i \to Q^j$ for every path of length $\leqslant 6$ from Q^i to Q^j in that picture.

Exercise 5.2.1.18. Let $c \in \mathrm{Ob}(\mathcal{C})$ be an object and let $I \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$ be another object. Consider c also as a functor $c\colon \underline{1} \to \mathcal{C}$ and recall the pullback functor $\Delta_c\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathbf{Set}$ and its left adjoint $\Sigma_c\colon \mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$ from Section 5.1.4.

a.) What is the set $\Delta_c(I)$?

b.) What is $\mathrm{Hom}_{\mathbf{Set}}(\{☺\}, \Delta_c(I))$?

c.) What is $\mathrm{Hom}_{\mathcal{C}\text{–}\mathbf{Set}}(\Sigma_c(\{☺\}), I)$?

d.) How does $\Sigma_c(\{☺\})$ compare to Y_c, the functor represented by c, as objects in $\mathcal{C}$–**Set**?

◊

Lemma 5.2.1.19 (Yoneda's lemma, part 2)**.** *Let $\mathcal{C}$ be a category. The assignment $c \mapsto Y_c$ from Lemma 5.2.1.13 extends to a functor $Y\colon \mathcal{C}^{\mathrm{op}} \to \mathcal{C}\text{–}\mathbf{Set}$, and this functor is fully faithful.*

*In particular, if $c, c' \in \mathrm{Ob}(\mathcal{C})$ are objects and there is an isomorphism $Y_c \cong Y_{c'}$ in $\mathcal{C}$–**Set**, then there is an isomorphism $c \cong c'$ in $\mathcal{C}$.*

Proof. See [Mac]. □

Exercise 5.2.1.20. The distributive law for addition of natural numbers says $(a+b) \times c = a \times c + b \times c$. Below we will give a proof of the distributive law, using category-theoretic reasoning. Annotate anything in red ink with a justification for why it is true.

Proposition 5.2.1.21. *For any natural numbers $a, b, c \in \mathbb{N}$, the distributive law*

$$(a+b)c = ac + bc$$

holds.

Sketch of proof. To finish, justify red stuff.
Let A, B, C be finite sets and let X be another finite set.

$$\begin{aligned}
\mathrm{Hom}_{\mathbf{Set}}((A+B) \times C, X) &\cong \mathrm{Hom}_{\mathbf{Set}}(A+B, X^C)\\
&\cong \mathrm{Hom}_{\mathbf{Set}}(A, X^C) \times \mathrm{Hom}_{\mathbf{Set}}(B, X^C)\\
&\cong \mathrm{Hom}_{\mathbf{Set}}(A \times C, X) \times \mathrm{Hom}_{\mathbf{Set}}(B \times C, X)\\
&\cong \mathrm{Hom}_{\mathbf{Set}}((A \times C) + (B \times C), X).
\end{aligned}$$

By the appropriate application of Yoneda's lemma, we see that there is an isomorphism

$$(A+B) \times C \cong (A \times C) + (B \times C)$$

in **Fin**. The result about natural numbers follows. □

◊

5.2.1.22 The subobject classifier $\Omega \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$

If $\mathcal{C}$ is a category then the functor category $\mathcal{C}$–**Set** is a very nice kind of category, called a *topos*. Note that when $\mathcal{C} = \underline{1}$ is the terminal category, then we have an isomorphism $\mathcal{C}\text{–}\mathbf{Set} \cong \mathbf{Set}$, so the category of sets is a special case of a topos. What is so interesting about toposes (or topoi) is that they so nicely generalize many properties of **Set**. In this short section we investigate only one such property, namely that $\mathcal{C}$–**Set** has a subobject classifier, denoted $\Omega \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$. In the case $\mathcal{C} = \underline{1}$, we saw back in Section 2.7.4.9 that the subobject classifier is $\{True, False\} \in \mathrm{Ob}(\mathbf{Set})$.

As usual, we consider the matter of subobject classifiers by grounding the discussion in terms of databases.

Definition 5.2.1.23. Let $\mathcal{C}$ be a category, let $\mathcal{C}$–**Set** denote its category of instances, and let $1 \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$ denote the terminal object. A *subobject classifier for* $\mathcal{C}$–**Set** is an object $\Omega_{\mathcal{C}} \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$ and a morphism $t\colon 1 \to \Omega_{\mathcal{C}}$ with the following property. For any monomorphism $f\colon X \to Y$ in $\mathcal{C}$–**Set**, there exists a unique morphism $char(f)\colon Y \to \Omega_{\mathcal{C}}$ such that the following diagram is a pullback in $\mathcal{C}$–**Set**:

$$\begin{tikzcd}
X \arrow[r, "!"] \arrow[d, "f"'] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[d, "t"] \\
Y \arrow[r, "char(f)"'] & \Omega_{\mathcal{C}}
\end{tikzcd}$$

In terms of databases, what this means is that for every schema $\mathcal{C}$ there is some special instance $\Omega_{\mathcal{C}} \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$ that somehow classifies sub-instances. When our schema is the terminal category, $\mathcal{C} = \underline{1}$, instances are sets and we saw in Definition 2.7.4.9 that the subobject classifier is $\Omega_{\underline{1}} = \{True, False\}$. One might think that the subobject classifier for $\mathcal{C}$–**Set** should just consist of a two-element set table-by-table, i.e. that for every $c \in \mathrm{Ob}(\mathcal{C})$ we should have $\Omega_{\mathcal{C}} =^? \{True, False\}$, but this is not correct.

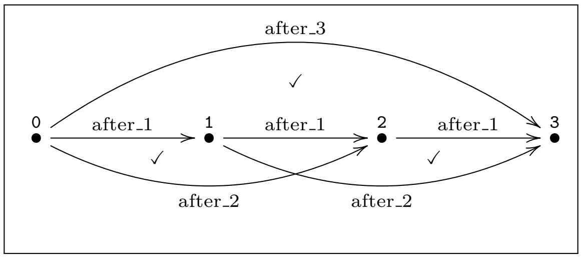
In fact, for any object $c \in \mathrm{Ob}(\mathcal{C})$, it is easy to say what $\Omega_{\mathcal{C}}(c)$ should be. We know by Yoneda's lemma (Lemma 5.2.1.13) that $\Omega_{\mathcal{C}}(c) = \mathrm{Hom}_{\mathcal{C}\text{–}\mathbf{Set}}(Y_c, \Omega_{\mathcal{C}})$, where Y_c is the functor represented by c. There is a bijection between $\mathrm{Hom}_{\mathcal{C}\text{–}\mathbf{Set}}(Y_c, \Omega_{\mathcal{C}})$ and the set of sub-instances of Y_c. Each morphism $f\colon c \to d$ in $\mathcal{C}$ induces a morphism $Y_f\colon Y_d \to Y_c$, and the map $\Omega_{\mathcal{C}}(f)\colon \Omega_{\mathcal{C}}(c) \to \Omega_{\mathcal{C}}(d)$ sends a sub-instance $A \subseteq Y_c$ to the pullback

$$\begin{tikzcd}
Y_f^{-1}(A) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & A \arrow[d] \\
Y_d \arrow[r, "Y_f"'] & Y_c
\end{tikzcd}$$

But this is all very abstract. We now give an example of a subobject classifier.

Example 5.2.1.24. Consider the category $\mathcal{C} \cong [3]$ depicted below

$\mathcal{C} :=$



To write down $\Omega_{\mathcal{C}}$ we need to understand the representable functors $Y_c \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$, for $c = \mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}$, as well as their subobjects. Here is $Y_{\mathtt{0}}$ as an instance:

$Y_{\mathtt{0}}(\mathtt{0})$			
ID	**after_1**	**after_2**	**after_3**
☺	after_1(☺)	after_2(☺)	after_3(☺)

$Y_{\mathtt{0}}(\mathtt{1})$		
ID	**after_1**	**after_2**
after_1(☺)	after_2(☺)	after_3(☺)

$Y_{\mathtt{0}}(\mathtt{2})$	
ID	**after_1**
after_2(☺)	after_3(☺)

$Y_{\mathtt{0}}(\mathtt{3})$
ID
after_3(☺)

What are the sub-instances of this? There is the empty sub-instance $\varnothing \subseteq Y_{\mathtt{0}}$ and the identity sub-instance $Y_{\mathtt{0}} \subseteq Y_{\mathtt{0}}$. But there are three more as well. Note that if we want to keep the ☺ row of table $\mathtt{0}$ then we have to keep everything. But if we throw away the ☺ row of table $\mathtt{0}$ we can still keep the rest and get a sub-instance. If we want to keep the after_1(☺) row of table $\mathtt{1}$ then we have to keep its images in tables $\mathtt{2}$ and $\mathtt{3}$. But we could throw away both the ☺ row of table $\mathtt{0}$ and the after_1(☺) row of table $\mathtt{1}$ and still keep the rest. And so on. In other words, the subobjects of $Y_{\mathtt{0}}$ are in bijection with the set $\Omega_{\mathcal{C}}(\mathtt{0}) := \{\mathit{yes}, \mathit{in\ 1}, \mathit{in\ 2}, \mathit{in\ 3}, \mathit{never}\}$.

The same analysis holds for the other tables of $\Omega_{\mathcal{C}}$. It looks like this:

$\Omega_{\mathcal{C}}(\mathtt{0})$			
ID	**after_1**	**after_2**	**after_3**
yes	*yes*	*yes*	*yes*
in 1	*yes*	*yes*	*yes*
in 2	*in 1*	*yes*	*yes*
in 3	*in 2*	*in 1*	*yes*
never	*never*	*never*	*never*

$\Omega_{\mathcal{C}}(\mathtt{1})$		
ID	**after_1**	**after_2**
yes	*yes*	*yes*
in 1	*yes*	*yes*
in 2	*in 1*	*yes*
never	*never*	*never*

$\Omega_{\mathcal{C}}(\mathtt{2})$	
ID	**after_1**
yes	*yes*
in 1	*yes*
never	*never*

$\Omega_{\mathcal{C}}(\mathtt{3})$
ID
yes
never

The morphism $1 \to \Omega_{\mathcal{C}}$ picks out the *yes* row of every table.

Now that we have constructed $\Omega_{\mathcal{C}} \in \mathrm{Ob}(\mathcal{C}\text{–}\mathbf{Set})$, we are ready to see it in action. What makes $\Omega_{\mathcal{C}}$ special is that for any instance $X \colon \mathcal{C} \to \mathbf{Set}$, the subinstances if X are in one-to-one correspondence with the morphisms $X \to \Omega_{\mathcal{C}}$. Consider the following arbitrary instance X, where the blue rows denote a sub-instance $A \subseteq X$.

$X(\mathtt{0})$			
ID	**after 1**	**after 2**	**after 3**
a_1	b_1	c_1	d_1
a_2	b_2	c_1	d_1
a_3	b_2	c_1	d_1
a_4	b_3	c_2	d_2
a_5	b_5	c_3	d_1

$X(\mathtt{1})$		
ID	**after 1**	**after 2**
b_1	c_1	d_1
b_2	c_1	d_1
b_3	c_2	d_2
b_4	c_1	d_1
b_5	c_3	d_1

$X(\mathtt{2})$	
ID	**after 1**
c_1	d_1
c_2	d_2
c_3	d_1

$X(\mathtt{3})$
ID
d_1
d_2

(5.6)

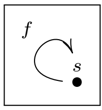
This blue sub-instance $A \subseteq X$ corresponds to a map $char(A)\colon X \to \Omega_{\mathcal{C}}$. That is for each $c \in \mathrm{Ob}(\mathcal{C})$ the rows in the c-table of X are sent to the rows in the c-table of $\Omega_{\mathcal{C}}$. The way $char(A)$ works is as follows. For each table i and row $x \in X(i)$, find the first column f in which the entry is blue (i.e. $f(x) \in A$), and send x to the corresponding element of $\Omega_{\mathcal{C}}(i)$. For example, $char(A)(\mathtt{0})$ sends a_1 to *in 2* and sends a_4 to *never*, and $char(A)(\mathtt{2})$ sends c_1 to *yes* and sends c_2 to *never*.

Exercise 5.2.1.25. a.) Write out the blue subinstance $A \subseteq X$ shown in (5.6) as an instance of $\mathcal{C}$, i.e. as four tables.

b.) This subinstance $A \subseteq X$ corresponds to a map $\ell := char(A)\colon X \to \Omega_{\mathcal{C}}$. For all $c \in \mathrm{Ob}(\mathcal{C})$ we have a function $\ell(c)\colon X(c) \to \Omega_{\mathcal{C}}(c)$. With $c = \mathtt{1}$, write out $\ell(\mathtt{1})\colon X(\mathtt{1}) \to \Omega_{\mathcal{C}}(\mathtt{1})$.

◊

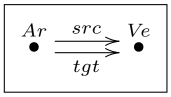
Exercise 5.2.1.26. Let $\mathcal{L}oop$ be the loop schema

$\mathcal{L}oop =$ .

a.) What is the subobject classifier $\Omega_{\mathcal{L}oop} \in \mathrm{Ob}(\mathcal{L}oop\text{–}\mathbf{Set})$?

b.) How does $\Omega_{\mathcal{L}oop}$ compare to the representable functor Y_s?

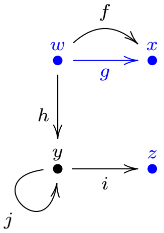
◊

Exercise 5.2.1.27. Let $\mathbf{GrIn} =$  be the indexing category for graphs.

a.) Write down the subobject classifier $\Omega_{\mathbf{GrIn}} \in \mathrm{Ob}(\mathbf{GrIn}\text{–}\mathbf{Set})$ in tabular form, i.e. as two tables.

b.) Draw $\Omega_{\mathbf{GrIn}}$ as a graph.

c.) Let G be the graph below and $G' \subseteq G$ the blue part.



Write down $G \in \mathrm{Ob}(\mathbf{GrIn}\text{–}\mathbf{Set})$ in tabular form.

d.) Write down the components of the natural transformation $char(G')\colon G \to \Omega_{\mathbf{GrIn}}$.

◊

5.2.2 Database instances in other categories

5.2.2.1 Representations of groups

The classical mathematical subject of *representation theory* is the study of $\mathrm{Fun}(G, \mathbf{Vect})$ where G is a group and $\mathbf{Vect}$ is the category of vector spaces (over say $\mathbb{R}$). Every such functor $F\colon G \to \mathbf{Vect}$ is called a *representation of* G. Since G is a category with one object $\blacktriangle$, F consists of a single vector space $V = F(\blacktriangle)$ together with an action of G on it.

We can think of this in terms of databases if we have a presentation of G in terms of generators and relations. The schema corresponding to G has one table and this table has a column for each generator. Giving a representation F is the same as giving an instance on our schema, with some properties that stem from the fact that our target category is **Vect** rather than **Set**. There are many possibilities for expressing [10] such data.

One possibility is if we could somehow draw V, say if V is 1-, 2-, or 3-dimensional. If so, let P be our chosen picture of V, e.g. P is the standard drawing of a Cartesian coordinate plane. Then every column of our table would consist entirely of the picture P instead of a set of rows. Drawing a point in the ID-column picture would result in a point being drawn in each other column's picture, in accordance with the G-action. Each column would of course respect addition and scalar multiplication.

Another possibility is to use the fact that there is a functor $U\colon \mathbf{Vect} \to \mathbf{Set}$, so our instance $F\colon G \to \mathbf{Vect}$ can be converted to an ordinary instance $U \circ F\colon G \to \mathbf{Set}$. We would have an ordinary set of rows. This set would generally be infinite, but it would be structured by addition and scalar multiplication. For example, assuming V is finite dimensional, one could find a few rows that generated the rest.

A third possibility is to use monads, which allow the table to have only as many rows as V has dimensions. This is a considerable savings of space. See Section 5.3.

5.2.2.2 Representations of quivers

Representation theory also studies representations of quivers. A *quiver* is just the free category (see Example 4.1.2.30) on a graph. If P is a graph with free category $\mathcal{P}$ then a representation of the quiver $\mathcal{P}$ is a functor $F\colon \mathcal{P} \to \mathbf{Vect}$. Such a representation consists of a vector space at every vertex of P and a linear transformation for every arrow. All of the discussion from Section 5.2.2.1 works in this setting, except that there is more than one table.

5.2.2.3 Other target categories

One can imagine the value of using target categories other than **Set** or **Vect** for databases.

Application 5.2.2.4. Geographic data consists of maps of the earth together with various functions on it. For example for any point on the earth one may want to know the average temperature recorded in the past 10 years, or the precise temperature at this moment. Earth can be considered as a topological space, E. Similarly, temperatures on earth reside on a continuum, say the space T of real numbers $[-100, 200]$. Thus the temperature record is a function $E \to T$.

Other records such as precipitation, population density, elevation, etc. can all be considered as continuous functions from E to some space. Agencies like the US Geological Survey hold databases of such information. By modeling them on functors $\mathcal{C} \to \mathbf{Top}$, they may be able to employ mathematical tools such as persistent homology [WeS] to find interesting invariants of the data.

◊◊

[10] We would use the term “representing” or ”presenting”, but they are both taken in the context of our narrative!

Application 5.2.2.5. Many other scientific disciplines could use the same kind of tool. For example, in studying the mechanics of materials, one may want to consider the material as a topological space M and measure values such as energy as a continuous $M \to E$. Such observations could be modeled by databases with target category **Top** or **Vect** rather than **Set**.

$\Diamond\Diamond$

5.2.3 Sheaves

Let X be a topological space (see Example 4.2.3.1), such as a sphere. In Section 5.2.2.3 we discussed continuous functions out of X, and their use in science (e.g. recording temperatures on the earth as a continuous map $X \to [-100, 200]$). Sheaves allow us to consider the local-global nature of such maps, taking into account reparable discrepancies in data gathering tools.

Application 5.2.3.1. Suppose that X is the topological space corresponding to the earth; by a *region* we mean an open subset $U \subseteq X$. Suppose that we cover X with 10,000 regions $U_1, U_2, \ldots, U_{10000}$, such that some of the regions overlap in a non-empty subregion (e.g. perhaps $U_5 \cap U_9 \neq \varnothing$). For each i, j let $U_{i,j} = U_i \cap U_j$.

For each region $U_i \subseteq X$ we have a temperature recording device, which gives a function $T_i \colon U_i \to [-100, 200]$. If $U_i \cap U_j \neq \varnothing$ then two different recording devices give us temperature data for the intersection $U_{i,j}$. Suppose we find that they do not give precisely the same data, but that there is a translation formula between their results. For example, T_i might register 3° warmer than T_j registers, throughout the region $U_i \cap U_j$.

A consistent system of translation formulas is called a *sheaf*. It does not demand a universal "true" temperature function, but only a consistent translation system between them.

$\Diamond\Diamond$

The following definitions (Definitions 5.2.3.2, 5.2.3.5) make the notion of sheaf precise, but we must go slowly (because it will already feel quick to the novice). For every region U, we can record the value of some function (say temperature) throughout U; although this record might consist of a mountain of data (a temperature for each point in U!), we think of it as one thing. That is, it is one element in the set of value-assignments throughout U. A sheaf holds the set of possible values-assignments-throughout-U's for all the different regions U, as well as how a value-assignment-throughout-U restricts to a value-assignment-throughout-V for any subset $V \subseteq U$.

Definition 5.2.3.2. Let X be a topological space, let $\text{Open}(X)$ denote its partial order of open sets, and let $\text{Open}(X)^{\text{op}}$ be the opposite category. A *presheaf on X* is a functor $\mathcal{O} \colon \text{Open}(X)^{\text{op}} \to \textbf{Set}$. For every open set $U \subseteq X$ we refer to the set $\mathcal{O}(U)$ as the *set of values-assignments throughout U of $\mathcal{O}$*. If $V \subseteq U$ is an open subset, it corresponds to an arrow in $\text{Open}(X)$ and applying the functor $\mathcal{O}$ yields a function called the *restriction map from U to V* and denoted $\rho_{V,U} \colon \mathcal{O}(U) \to \mathcal{O}(V)$. Given $a \in \mathcal{O}(U)$, we may denote $\rho_{V,U}(a)$ by $a|_V$; it is called *the restriction of a to V*.

The *category of presheaves on X* is simply $\text{Open}(X)^{\text{op}}$–**Set**; see Definition 4.3.3.1.

Exercise 5.2.3.3.

a.) Come up with 4 overlapping open subsets that cover the square $X := [0,3] \times [0,3] \subseteq \mathbb{R}^2$. Write down a label for each open set as well as a label for each overlap (2-fold, 3-fold, etc.); you now have labeled n open sets. For each of these open sets, draw a dot with the appropriate label, and then draw an arrow from one dot to another when the first refers to an open subset of the second. This is a preorder; call it Open(X). Now make up and write down formulas $R_1 \colon X \to \mathbb{R}$ and $R_2 \colon X \to \mathbb{R}$ with $R_1 \leqslant R_2$, expressing a range of temperatures $R_1(p) \leqslant x \leqslant R_2(p)$ that an imaginary experiment shows can exist at each point p in the square.

b.) Suppose we now tried to make our presheaf $\mathcal{O} \colon \mathrm{Open}(X)^{\mathrm{op}} \to \mathbf{Set}$ as follows. For each of your open sets, say A, we could put

$$\mathcal{O}(A) := \{f \colon A \to \mathbb{R} \mid R_1(a) \leqslant f(a) \leqslant R_2(a)\}.$$

What are the restriction maps? Do you like the name "value-assignment throughout A" for elements of $\mathcal{O}(A)$?

c.) We can now make another presheaf $\mathcal{O}'$ given the same experiment. For each of your open sets, say A, we could put

$$\mathcal{O}'(A) := \{f \colon A \to \mathbb{R} \mid f \text{ is continuous, and } R_1(a) \leqslant f(a) \leqslant R_2(a)\}.$$

Are you comfortable with the idea that there is a morphism of presheaves $\mathcal{O}' \to \mathcal{O}$?

◊

Before we define sheaves, we need to clarify the notion of covering. Suppose that U is a region and that $V_1, \ldots, V_n$ are subregions (i.e. for each $1 \leqslant i \leqslant n$ we have $V_i \subseteq U$). Then we say that the V_i *cover* U if every point in U is in V_i for some i. Another way to say this is that the natural function $\sqcup_i V_i \to U$ is surjective.

Example 5.2.3.4. Let $X = \mathbb{R}$ be the space of real numbers, and define the following open subsets: $U = (5, 10), V_1 = (5, 7), V_2 = (6, 9), V_3 = (7, 10)$. [11] Then V_1, V_2, V_3 is a cover of U. It has overlaps $V_{12} = V_1 \cap V_2 = (6, 7)$, $V_{13} = V_1 \cap V_3 = \varnothing$, $V_{23} = V_2 \cap V_3 = (7, 9)$.

Given a presheaf $\mathcal{O} \colon \mathrm{Open}(X)^{\mathrm{op}} \to \mathbf{Set}$, we have sets and functions as in the following (incomplete) diagram

$$\begin{tikzcd}
 & \mathcal{O}(V_1) \arrow[dr] & \\
 & & \mathcal{O}(V_1 \cap V_2) \\
\mathcal{O}(U) \arrow[uur] \arrow[r] \arrow[ddr] & \mathcal{O}(V_2) \arrow[ur] \arrow[dr] & \\
 & & \mathcal{O}(V_2 \cap V_3) \\
 & \mathcal{O}(V_3) \arrow[ur] &
\end{tikzcd}$$

A presheaf $\mathcal{O}$ on X tells us what value-assignments throughout U can exist for each U. Suppose we have a value-assignment $a \in \mathcal{O}(U)$ throughout U and another value-assignment $a' \in \mathcal{O}(U')$ throughout U', and suppose that they agree as value-assignments throughout $U \cap U'$, i.e. $a|_{U \cap U'} = a'|_{U \cap U'}$. In this case we should have a unique value-assignment $b \in \mathcal{O}(U \cup U')$ throughout $U \cup U'$ that agrees on the U-part with a and agrees on the U'-part with a'; i.e. $b|_U = a$ and $b|_{U'} = a'$. This is the sheaf condition.

[11] We use parentheses to denote open intervals of real numbers. For example $(6, 9)$ denotes the set $\{x \in \mathbb{R} \mid 6 < x < 9\}$.

Definition 5.2.3.5. Let X be a topological space, let $\text{Open}(X)$ be its partial order of open sets, and let $\mathcal{O}\colon \text{Open}(X)^{\text{op}} \to \textbf{Set}$ be a presheaf. Given an open set $U \subseteq X$ and a cover $V_1, \ldots, V_n$ of U, the following condition is called the *sheaf condition* for that cover.

Sheaf condition Given a sequence $a_1, \ldots, a_n$ where each is a value-assignment $a_i \in \mathcal{O}(V_i)$ throughout V_i, suppose that for all i, j we have $a_i|_{V_i \cap V_j} = a_j|_{V_i \cap V_j}$; then there is a unique value-assignment $b \in \mathcal{O}(U)$ such that $b|_{V_i} = a_i$.

The presheaf $\mathcal{O}$ is called a *sheaf* if it satisfies the sheaf condition for every cover.

Example 5.2.3.6. Let $X = \mathbb{R}$ and let U, V_1, V_2, V_3 be the open cover given in Example 5.2.3.4. Given a measurement taken throughout V_1, a measurement taken throughout V_2, and a measurement taken throughout V_3, we have elements $a_1 \in \mathcal{O}(V_1), a_2 \in \mathcal{O}(V_2)$, and $a_3 \in \mathcal{O}(V_3)$. If they are in agreement on the overlap intervals, we can *glue* them to give a measurement throughout U.

Remark 5.2.3.7. In Application 5.2.3.1, we said that sheaves would help us patch together information from different sources. Even if different temperature-recording devices T_i and T_j registered different temperatures on an overlapping region $U_i \cap U_j$, we said they could be patched together if there was a consistent translation system between their results. What is actually needed is a set of isomorphisms

$$p_{i,j}\colon T_i|_{U_{i,j}} \xrightarrow{\cong} T_j|_{U_{i,j}}$$

that translate between them, and that these $p_{i,j}$'s act in concert with one another. This (when precisely defined,) is called *descent data*.. The way it interacts with our definition of sheaf given in Definitions 5.2.3.2 and 5.2.3.5 is buried in the restriction maps ρ for the overlaps as subsets $U_{i,j} \subseteq U_i$ and $U_{i,j} \subseteq U_j$. We will not explain further here. One can see [Gro].

Application 5.2.3.8. Consider outer space as a topological space X. Different astronomers record observations. Let $C = [390, 700]$ denote the set of wavelengths in the visible light spectrum (written in nanometers). Given an open subset $U \subseteq X$ let $\mathcal{O}(U)$ denote the set of functions $U \to C$. The presheaf $\mathcal{O}$ satisfies the sheaf condition; this is the taken-for-granted fact that we can patch together different observations of space.

Below are three views of the night sky. Given a telescope position to obtain the first view, one moves the telescope right and a little down to obtain the second and one moves it down and left to obtain the third. [12]

[12] Image credit: NASA, ESA, Digitized Sky Survey Consortium.

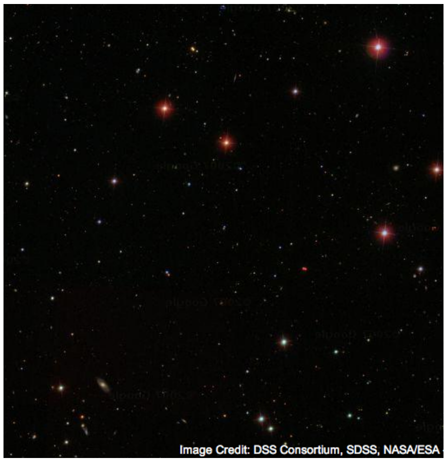


Image Credit: DSS Consortium, SDSS, NASA/ESA

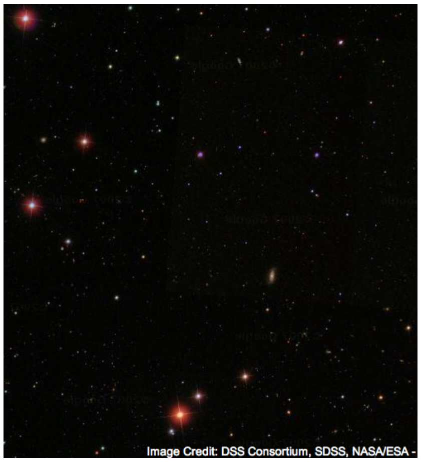


Image Credit: DSS Consortium, SDSS, NASA/ESA -

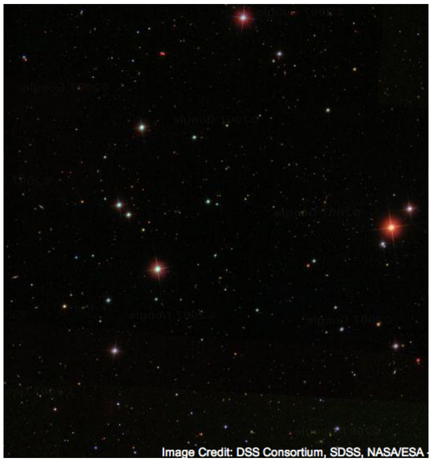


Image Credit: DSS Consortium, SDSS, NASA/ESA

These are value-assignments $a_1 \in \mathcal{O}(V_1), a_2 \in \mathcal{O}(V_2)$, and $a_3 \in \mathcal{O}(V_3)$ throughout subsets $V_1, V_2, V_3 \subseteq X$ (respectively). These subsets V_1, V_2, V_3 cover some (strangely-shaped) subset $U \subseteq X$. The sheaf condition says that these three value-assignments glue together to form a single value-assignment throughout U:

◊◊

Exercise 5.2.3.9. Find an application of sheaves in your own domain of expertise. ◊

Application 5.2.3.10. Suppose we have a sheaf for temperatures on earth. For every region U we have a set of theoretically possible temperature-assignments throughout U. For example we may know that if it is warm in Texas, warm in Arkansas, and warm in Kansas, then it cannot be cold in Oklahoma. With such a sheaf $\mathcal{O}$ in hand, one can use facts about the temperature in one region U to predict the temperature in another region V.

The mathematics is as follows. Suppose given regions $U, V \subseteq X$ and a subset $A \subseteq \mathcal{O}(U)$ corresponding to what we know about the temperature assignment throughout U. We take the following fiber product

$$\begin{tikzcd}
(\rho_{U,X})^{-1}(A) \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathcal{O}(X) \arrow[r, "\rho_{V,X}"] \arrow[d, "\rho_{U,X}"] & \mathcal{O}(V) \\
A \arrow[r] & \mathcal{O}(U) &
\end{tikzcd}$$

The image of the top map is a subset of $\mathcal{O}(V)$ telling us which temperature-assignments are possible throughout V given our knowledge A about the temperature throughout U.

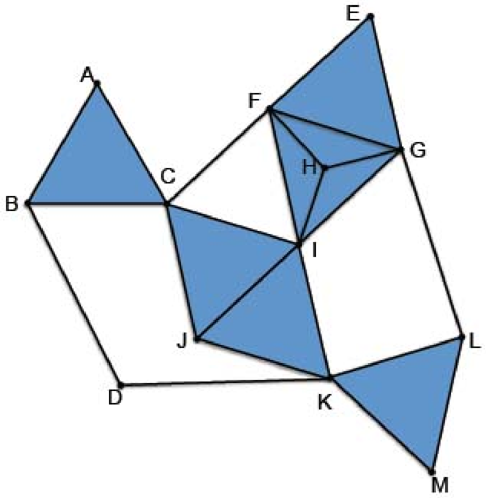
We can imagine the same type of prediction systems for other domains as well, such as the energy of various parts of a material. ◊◊

Example 5.2.3.11. In Exercises 4.2.4.3 and 4.2.4.4 we discussed the idea of laws being dictated or respected throughout a jurisdiction. If X is earth, to every jurisdiction $U \subseteq X$ we assign the set $\mathcal{O}(U)$ of laws that are dictated to hold throughout U. Given a law on U and a law on V, we can see if they amount to the same law on $U \cap V$. For example, on U a law might say “no hunting near rivers” and on V a law might say “no hunting in public areas”. It just so happens that on $U \cap V$ all public areas are near rivers and vice versa, so the laws agree there. These laws patch together to form a single rule about hunting that is enforced throughout the union $U \cup V$, respected by all jurisdictions within it.

5.2.3.12 Sheaf of ologged concepts

Definition 5.2.3.5 defines what should be called a sheaf of sets. We can discuss sheaves of groups or even sheaves of categories. Here is an application of the latter.

Recall the notion of simplicial complexes discussed in Section 2.7.4.3. They look like this:



(5.7)

Given such a simplicial complex X, we can imagine each vertex $v \in X_0$ as an entity with a worldview (e.g. a person) and each simplex as the common worldview shared by its vertices. To model this, we will assign to each vertex $v \in X$ an olog $\mathcal{O}(v)$, corresponding to the worldview held by that entity, and to each simplex $u \in X_n$, we assign an olog $\mathcal{O}(u)$ corresponding to a *common ground* worldview.. Recall that X is a subset of $\mathbb{P}(X_0)$; it is a preorder and its elements (the simplices) are ordered by inclusion. If u, v are simplices with $u \subseteq v$ then we want a map of ologs (i.e. a schema morphism) $\mathcal{O}(v) \to \mathcal{O}(u)$ corresponding to how any idea that is shared among the people in v is shared among the people in u. Thus we have a functor $\mathcal{O}\colon X \to \mathbf{Sch}$ (where we are forgetting the distinction between ologs and databases for notational convenience).

To every simplicial complex (indeed every ordered set) one can associate a topological space; in fact we have a functor $Alx\colon \mathbf{PrO} \to \mathbf{Top}$, called the Alexandrov functor. Applying $Alx(X^{\text{op}})$ we have a space which we denote by $\mathcal{X}$. One can visualize $\mathcal{X}$ as X, but the open sets include unions of simplices. There is a unique sheaf of categories on $\mathcal{X}$ that behaves like X on simplices.

How does this work in the case of our sheaf $\mathcal{O}$ of worldviews? For simplices such as (A) or (CI), the sheaf returns the olog corresponding to that person or shared worldview. But for open sets like the union of (CIJ) and (IJK), what we get is the olog consisting of the types shared by C, I, and J for which I and J affirm agreement with types shared by I, J, and K.

Example 5.2.3.13. Imagine two groups of people G_1 and G_2 each making observations about the world. Suppose that there is some overlap $H = G_1 \cap G_2$. Then it may happen that there is a conversation including G_1 and G_2 and both groups are talking about something and, although using different words, H says “you guys are talking about the same things, you just use different words." In this case there is an object-assignment throughout $G_1 \cup G_2$ that agrees with both those on G_1 and those on G_2.

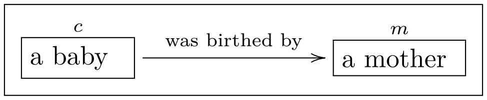
5.2.3.14 Time

One can use sheaves to model objects in time; Goguen gave an approach to this in [Gog]. For another approach, let $\mathcal{C}$ be a database schema. The lifespan of information about the world is generally finite; that is, what was true yesterday is not always the case today. Thus we can associate to each interval U of time the information that we deem to hold throughout U. This is sometimes called the *valid time* of the data.

If something is the case throughout U and we have a subset $V \subseteq U$ then of course it is the case throughout V. And the sheaf condition holds too: if some information holds throughout U and some other information holds throughout U', and if these two things restrict to the same information on the overlap $U \cap V$, then they can be glued to information that holds throughout the union $U \cup V$.

So we can model information-change over time by using a sheaf of $\mathcal{C}$-sets on the topological space $\mathbb{R}$. One way to think of this is simply as an instance on the schema $\mathcal{C} \times \text{Open}(\mathbb{R})^{\text{op}}$. The sheaf condition is just an added property that our instances have to obey.

Example 5.2.3.15. Consider a hospital in which babies are born. In our scenario, mothers enter the hospital, babies are born, mothers and babies leave the hospital. Let $\mathcal{C}$ be the schema



Consider the 8-hour intervals

$$\begin{aligned}
\text{Shift}_1 &:= (\text{Jan } 1 - 00:00,\ \text{Jan } 1 - 08:00),\\
\text{Shift}_2 &:= (\text{Jan } 1 - 04:00,\ \text{Jan } 1 - 12:00),\\
\text{Shift}_3 &:= (\text{Jan } 1 - 8:00,\ \text{Jan } 1 - 16:00).
\end{aligned}$$

The nurses take shifts of 8 hours, overlapping with their predecessors by 4 hours, and they record in the database only patients that were there throughout their shift or throughout any overlapping shift. A mother might be in the hospital throughout shift 1, arriving before the new year. A baby is born at 05:00 on Jan 1, and thus does not make it into the Shift_1-table, but does make it into the $(\text{Shift}_1 \cap \text{Shift}_2)$-table. The two are there until 17:00 on Jan 1, and so they are recorded in the Shift_2 and Shift_3 tables.

Whether or not this implementation of the sheaf semantics is most useful in practice is certainly debatable. But something like this could easily be useful as a semantics, i.e. a way of thinking about, the temporal nature of data.

5.3 Monads

Monads would probably not have been invented without category theory, but they have been quite useful in formalizing algebra, calculating invariants of topological spaces, and imbedding non-functional operations into functional programming languages. We will mainly discuss monads in terms of how they can help us make modeling contexts explicit, and in so doing allow us to simplify the language we use in the model.

Much of the following material on monads is taken from [Sp3].

5.3.1 Monads formalize context

Monads can formalize assumptions about the way one will do business throughout a domain. For example, suppose that we want to consider functions that do not have to return a value for all inputs. Such *partial functions* can be composed. Indeed, given a partial function $f\colon A \to B$ and a partial function $g\colon B \to C$, one gets a partial function $g \circ f\colon A \to C$ in an obvious way.

Here we are drawing arrows as though we are talking about functions, but there is an implicit context in which we are actually talking about partial functions. Monads allow us to write things in the "functional" way while holding the underlying context. What makes them useful is that the notion of *context* we are using here is made formal.

Example 5.3.1.1 (Partial functions). Partial functions can be modeled by ordinary functions, if we add a special "no answer" element to the codomain. That is, the set of partial functions $A \to B$ is in one-to-one correspondence with the set of ordinary functions $A \to B \sqcup \{☺\}$. For example, suppose we want to model the partial function $f(x) := \frac{1}{x^2-1}\colon \mathbb{R} \to \mathbb{R}$ in this way, we would use the function

$$f(x) := \begin{cases} \frac{1}{x^2-1} & \text{if } x \neq -1 \text{ and } x \neq 1, \\ ☺ & \text{if } x = -1, \\ ☺ & \text{if } x = 1. \end{cases}$$

An ordinary function $f\colon A \to B$ can be considered a partial function because we can compose with the inclusion

$$B \to B \sqcup \{☺\} \tag{5.8}$$

But how do we compose two partial functions written in this way? Suppose $f\colon A \to B \sqcup \{☺\}$ and $g\colon B \to C \sqcup \{☺\}$ are functions. First form a new function

$$g' := g \sqcup \{☺\}\colon B \sqcup \{☺\} \to C \sqcup \{☺\} \sqcup \{☺\} \tag{5.9}$$

then compose to get $(g' \circ f)\colon A \to C \sqcup \{☺\} \sqcup \{☺\}$, and finally send both ☺'s to the same element by composing with

$$C \sqcup \{☺\} \sqcup \{☺\} \to C \sqcup \{☺\}. \tag{5.10}$$

What does this mean? Every element $a \in A$ is sent by f to either an element $b \in B$ or "no answer". If it has an answer $f(a) \in B$, this is either sent by g to an element $g(f(a)) \in C$ or to "no answer". We get a partial function $A \to C$ by sending a to $g(f(a))$ if possible or to "no answer" if it gets stopped along the way.

This monad is sometimes called the *maybe monad* in computer science, because a partial function $f\colon A \to B$ takes every element of A and either outputs just an element of B or outputs nothing; more succinctly, it outputs a "maybe B".

Application 5.3.1.2. Experiments are supposed to be performed objectively, but suppose we imagine that changing the person who performs the experiment, say in psychology, may change the outcome. Let A be the set of experimenters, let X be the parameter space for the experimental variables (e.g. $X = \text{Age} \times \text{Income}$) and let Y be the observation space (e.g. $Y = \text{propensity for violence}$). Then whereas we want to think of such an experiment as telling us about a function $f\colon X \to Y$, we may want to make some of the context explicit by including information about who performed the experiment. That is, we are really finding a function $f\colon X \times A \to Y$.

However, it may be the case that even ascertaining someones age or income, which is done by asking that person, is subject to who in A is doing the asking, and so we again want to consider the experimenter as part of the equation. In this case, we can use a monad to hide the fact that everything in sight is assumed to be influenced by A. In other words, we want to announce once and for all our modeling context—that every observable is possibly influenced by the observer—so that it can recede into the background.

We will return to this in Examples 5.3.2.6 and 5.3.3.4.

◊◊

5.3.2 Definition and examples

What aspects of Example 5.3.1.1 are really about monads, and what aspects are just about partial functions in particular? It is a functor and a pair of natural transformations that showed up in (5.9), (5.8), and (5.10). In this section we will give the definition and a few examples. We will return to our story about how monads formalize context in Section 5.3.3.

Definition 5.3.2.1 (Monad)**.** A *monad on* **Set** is defined as follows: One announces some constituents (A. functor, B. unit map, C. multiplication map) and asserts that they conform to some laws (1. unit laws, 2. associativity law). Specifically, one announces

A. a functor $T\colon \mathbf{Set} \to \mathbf{Set}$,

B. a natural transformation $\eta\colon \mathrm{id}_{\mathbf{Set}} \to T$, and

C. a natural transformation $\mu\colon T \circ T \to T$

We sometimes refer to the functor T as though it were the whole monad; we call η the *unit map* and we call μ the *multiplication map*. One asserts that the following laws hold:

1. The following diagrams of functors **Set** → **Set** commute:

$$\begin{tikzcd} T \circ \mathrm{id}_{\mathbf{Set}} \arrow[r, "\mathrm{id}_T \diamond \eta"] \arrow[dr, "="'] & T \circ T \arrow[d, "\mu"] \\ & T \end{tikzcd} \qquad \begin{tikzcd} \mathrm{id}_{\mathbf{Set}} \circ T \arrow[r, "\eta \diamond \mathrm{id}_T"] \arrow[dr, "="'] & T \circ T \arrow[d, "\mu"] \\ & T \end{tikzcd}$$

2. The following diagram of functors **Set** → **Set** commutes:

$$\begin{tikzcd} T \circ T \circ T \arrow[r, "\mu \diamond \mathrm{id}_T"] \arrow[d, "\mathrm{id}_T \diamond \mu"'] & T \circ T \arrow[d, "\mu"] \\ T \circ T \arrow[r, "\mu"'] & T \end{tikzcd}$$

Example 5.3.2.2 (List monad)*.* We now go through Definition 5.3.2.1 using what is called the List monad. The first step is to give a functor $\mathrm{List}\colon \mathbf{Set} \to \mathbf{Set}$, which we did in Example 4.1.2.18. Recall that if $X = \{p, q, r\}$ then $\mathrm{List}(X)$ includes the empty list $[\ ]$, singleton lists, such as $[p]$, and any other list of elements in X, such as $[p, p, r, q, p]$. Given a function $f\colon X \to Y$, one obtains a function $\mathrm{List}(f)\colon \mathrm{List}(X) \to \mathrm{List}(Y)$ by entry-wise application of f.

As a monad, the functor List comes with two natural transformations, a unit map η and a multiplication map μ. Given a set X, the unit map $\eta_X\colon X \to \mathrm{List}(X)$ returns singleton lists as follows

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] & \mathrm{List}(X) \\
p \arrow[r, mapsto] & {[p]} \\
q \arrow[r, mapsto] & {[q]} \\
r \arrow[r, mapsto] & {[r]}
\end{tikzcd}$$

Given a set X, the multiplication map $\mu_X\colon \mathrm{List}(\mathrm{List}(X)) \to \mathrm{List}(X)$ flattens lists of lists as follows.

$$\begin{tikzcd}
\mathrm{List}(\mathrm{List}(X)) \arrow[r, "\mu_X"] & \mathrm{List}(X) \\
{\big[[q,p,r],[],[q,r,p,r],[r]\big]} \arrow[r, mapsto] & {[q,p,r,q,r,p,r,r]}
\end{tikzcd}$$

The naturality of η and μ just mean that these maps work appropriately well under term-by-term replacement by a function $f\colon X \to Y$. Finally the three monad laws from Definition 5.3.2.1 can be exemplified as follows:

$$\begin{tikzcd}
{[p,q,q]} \arrow[r, mapsto, "\mathrm{id}_{\mathrm{List}}\circ\eta"] \arrow[dr, equal] & {\big[[p],[q],[q]\big]} \arrow[d, mapsto, "\mu"] \\
& {[p,q,q]}
\end{tikzcd}
\qquad
\begin{tikzcd}
{[p,q,q]} \arrow[r, mapsto, "\eta\circ\mathrm{id}_{\mathrm{List}}"] \arrow[dr, equal] & {\big[[p,q,q]\big]} \arrow[d, mapsto, "\mu"] \\
& {[p,q,q]}
\end{tikzcd}$$

$$\begin{tikzcd}
{\Big[\big[[p,q],[r]\big],\big[[],[r,q,q]\big]\Big]} \arrow[r, mapsto, "\mu\circ\mathrm{id}_{\mathrm{List}}"] \arrow[d, mapsto, "\mathrm{id}_{\mathrm{List}}\circ\mu"'] & {\big[[p,q],[r],[],[r,q,q]\big]} \arrow[d, mapsto, "\mu"] \\
{\big[[p,q,r],[r,q,q]\big]} \arrow[r, mapsto, "\mu"'] & {[p,q,r,r,q,q]}
\end{tikzcd}$$

Exercise 5.3.2.3. Let $\mathbb{P}\colon \mathbf{Set} \to \mathbf{Set}$ be the powerset functor, so that given a function $f\colon X \to Y$ the function $\mathbb{P}(f)\colon \mathbb{P}(X) \to \mathbb{P}(Y)$ is given by taking images.

a.) Make sense of the following statement: "with η defined by singleton subsets and with μ defined by union, $\top := (\mathbb{P}, \eta, \mu)$ is a monad".

b.) With $X = \{a, b\}$, write down the function η_X as a 2-row, 2-column table, and write down the function μ_X as a 16-row, 2-column table (you can stop after 5 rows if you fully get it).

c.) Check that you believe the monad laws from Definition 5.3.2.1.

◊

Example 5.3.2.4 (Partial functions as a monad). Here is the monad for partial functions. The functor $T\colon \mathbf{Set} \to \mathbf{Set}$ sends a set X to the set $X \sqcup \{\smiley\}$. Clearly, given a function $f\colon X \to Y$ there is an induced function $f \sqcup \{\smiley\}\colon X \sqcup \{\smiley\} \to Y \sqcup \{\smiley\}$, so this is a functor. The natural transformation $\eta\colon \mathrm{id} \to T$ is given on a set X by the component function

$$\eta_X\colon X \to X \sqcup \{\smiley\}$$

that includes $X \hookrightarrow X \sqcup \{\smiley\}$. Finally, the natural transformation $\mu\colon T \circ T \to T$ is given on a set X by the component function

$$\mu_X\colon X \sqcup \{\smiley\} \sqcup \{\smiley\} \longrightarrow X \sqcup \{\smiley\}$$

that collapses both copies of $\smiley$.

Exercise 5.3.2.5. Let E be a set, elements we will refer to as *exceptions*. We imagine that a function $f\colon X \to Y$ either outputs a value or one of these exceptions, which might be things like “overflow!” or “division by zero!”, etc. Let $T\colon \mathbf{Set} \to \mathbf{Set}$ be the functor $X \mapsto X \sqcup E$. Follow Example 5.3.2.4 and come up with a unit map η and a multiplication map μ for which (T, η, μ) is a monad. ◊

Example 5.3.2.6. Fix a set A. Let $T\colon \mathbf{Set} \to \mathbf{Set}$ be given by $T(X) = X^A = \mathrm{Hom}_{\mathbf{Set}}(A, X)$; this is a functor. For a set X, let $\eta_X\colon X \to T(X)$ be given by the constant function, $x \mapsto c_x\colon A \to X$ where $c_x(a) = x$ for all $a \in A$. To specify a function

$$\mu_X\colon \mathrm{Hom}_{\mathbf{Set}}(A, T(X)) \to \mathrm{Hom}_{\mathbf{Set}}(A, X),$$

we curry and need a function $A \times \mathrm{Hom}_{\mathbf{Set}}(A, T(X)) \to X$. We have an evaluation function (see Exercise 2.7.2.5) $ev\colon A \times \mathrm{Hom}_{\mathbf{Set}}(A, T(X)) \to T(X)$, and we have an identity function $\mathrm{id}_A\colon A \to A$, so we have a function $(\mathrm{id}_A \times ev)\colon A \times \mathrm{Hom}_{\mathbf{Set}}(A, T(X)) \longrightarrow A \times T(X)$. Composing that with another evaluation function $A \times \mathrm{Hom}_{\mathbf{Set}}(A, X) \to X$ yields our desired μ_X. Namely, for all $b \in A$ and $f \in \mathrm{Hom}(A, T(X))$ we have

$$\mu_X(f)(b) = f(b)(b).$$

Remark 5.3.2.7. Monads can be defined on categories other than **Set**. In fact, for any category $\mathcal{C}$ one can take Definition 5.3.2.1 and replace every occurrence of **Set** with $\mathcal{C}$ and obtain the definition for monads on $\mathcal{C}$. We have actually seen a monad $(\mathrm{Paths}, \eta, \mu)$ on the category **Grph** of graphs before, namely in Examples 4.3.1.12 and 4.3.1.13. That is, $\mathrm{Paths}\colon \mathbf{Grph} \to \mathbf{Grph}$, which sends a graph to its paths-graph is the functor part. The unit map η includes a graph into its paths-graph using the observation that every arrow is a path of length 1. And the multiplication map μ concatenates paths of paths. The Kleisli category of this monad (see Definition 5.3.3.1) is used, e.g. in (4.14) to define morphisms of database schemas.

5.3.3 Kleisli category of a monad

Given a monad $\top := (T, \eta, \mu)$, we can form a new category $\mathbf{Kls}(\top)$.

Definition 5.3.3.1. Let $\top = (T, \eta, \mu)$ be a monad on **Set**. Form a new category, called the *Kleisli category for* $\top$, denoted $\mathbf{Kls}(\top)$, with sets as objects, $\mathrm{Ob}(\mathbf{Kls}(\top)) := \mathrm{Ob}(\mathbf{Set})$, and with

$$\mathrm{Hom}_{\mathbf{Kls}(\top)}(X, Y) := \mathrm{Hom}_{\mathbf{Set}}(X, T(Y))$$

for sets X, Y. The identity morphism $\mathrm{id}_X\colon X \to X$ in $\mathbf{Kls}(\top)$ is given by $\eta\colon X \to T(X)$ in $\mathbf{Set}$. The composition of morphisms $f\colon X \to Y$ and $g\colon Y \to Z$ in $\mathbf{Kls}(\top)$ is given as follows. Writing them as functions, we have $f\colon X \to T(Y)$ and $g\colon Y \to T(Z)$. The first step is to apply the functor T to g, giving $T(g)\colon T(Y) \to T(T(Z))$. Then compose with f to get $T(g) \circ f\colon X \to T(T(Z))$. Finally, compose with $\mu_Z\colon T(T(Z)) \to T(Z)$ to get the required function $X \to T(Z)$. The associativity of this composition formula follows from the associativity law for monads.

Example 5.3.3.2. Recall the monad $\top$ for partial functions, $T(X) = X \sqcup \{☺\}$, from Example 5.3.2.4. The Kleisli category $\mathbf{Kls}(\top)$ has sets as objects, but a morphism $f\colon X \to Y$ means a function $X \to Y \sqcup \{☺\}$, i.e a partial function. Given another morphism $g\colon Y \to Z$, the composition formula in $\mathbf{Kls}(\top)$ ensures that $g \circ f\colon X \to Z$ has the appropriate behavior.

Note how this monad allows us to make explicit our assumption that all functions are partial, and then hide it away from our notation.

Remark 5.3.3.3. For any monad $\top = (T, \eta, \mu)$ on $\mathbf{Set}$, there is a functor $i\colon \mathbf{Set} \to \mathbf{Kls}(\top)$ given as follows. On objects we have $\mathrm{Ob}(\mathbf{Kls}(\top)) = \mathrm{Ob}(\mathbf{Set})$, so take $i = \mathrm{id}_{\mathrm{Ob}(\mathbf{Set})}$. Given a morphism $f\colon X \to Y$ in $\mathbf{Set}$, we need a morphism $i(f)\colon X \to Y$ in $\mathbf{Kls}(\top)$, i.e. a function $i(f)\colon X \to T(Y)$. We assign $i(f)$ to be the composite $X \xrightarrow{f} Y \xrightarrow{\eta} T(Y)$. The functoriality of this mapping follows from the unit law for monads.

The point is that any ordinary function (morphism in $\mathbf{Set}$) has an interpretation as a morphism in the Kleisli category of any monad. More categorically, there is a functor $\mathbf{Set} \to \mathbf{Kls}(\top)$.

Example 5.3.3.4. In this example we return to the setting laid out by Application 5.3.1.2 where we had a set A of experimenters and assumed that the person doing the experiment may affect the outcome. We use the monad $\top = (T, \eta, \mu)$ from Example 5.3.2.6 and hope that $\mathbf{Kls}(\top)$ will conform to our understanding of how to manage the affect of the experimenter on data.

The objects of $\mathbf{Kls}(\top)$ are ordinary sets, but a map $f\colon X \to Y$ in $\mathbf{Kls}(\top)$ is a function $X \to Y^A$. By currying this is the same as a function $X \times A \to Y$, as desired. To compose f with $g\colon Y \to Z$ in $\mathbf{Kls}(\top)$, we follow the formula. It turns out to be equivalent to the following. We have a function $X \times A \to Y$ and a function $Y \times A \to Z$. Modifying the first slightly, we have a function $X \times A \to Y \times A$, by identity on A, and we can now compose to get $X \times A \to Z$.

What does this say in terms of experimenters affecting data gathering? It says that if we work within $\mathbf{Kls}(\top)$ then we will be able to assume that the experimenter is being taken into account; all proposed functions $X \to Y$ are actually functions $A \times X \to Y$. The natural way to compose these experiments is that we only consider the data from one experiment to feed into another if the experimenter is the same in both experiments. [13]

Exercise 5.3.3.5. In Exercise 5.3.2.3 we discussed the power set monad $\top = (\mathbb{P}, \eta, \mu)$.

a.) Can you find a way to relate the morphisms in $\mathbf{Kls}(\top)$ to relations? That is, given a morphism $f\colon A \to B$ in $\mathbf{Kls}(\top)$, is there a natural way to associate to it a relation $R \subseteq A \times B$?

[13] This requirement seems a bit stringent, but it can be mitigated in a variety of ways. One such way is to notice that by Remark 5.3.3.3 that we have not added any requirement, because any old way of doing business yields a valid new way of doing business (we just say "every experimenter would get the same result"). Another way would be to hand off the experiment results to another person, who could carry it forward (see Example 5.3.3.8).

b.) How does the composition formula in $\mathbf{Kls}(\top)$ relate to the composition of relations given in Definition 2.5.2.3? [14]

$\lozenge$

Exercise 5.3.3.6. Let $\top = (\mathbb{P}, \eta, \mu)$ be the power set monad. The category $\mathbf{Kls}(\top)$ is closed under binary products, i.e. every pair of objects $A, B \in \mathrm{Ob}(\mathbf{Kls}(\top))$ have a product in $\mathbf{Kls}(\top)$. What is the product of $A = \{1,2,3\}$ and $B = \{a,b\}$? $\lozenge$

Exercise 5.3.3.7. Let $\top = (\mathbb{P}, \eta, \mu)$ be the power set monad. The category $\mathbf{Kls}(\top)$ is closed under binary coproducts, i.e. every pair of objects $A, B \in \mathrm{Ob}(\mathbf{Kls}(\top))$ have a coproduct in $\mathbf{Kls}(\top)$. What is the coproduct of $A = \{1,2,3\}$ and $B = \{a,b\}$? $\lozenge$

Example 5.3.3.8. Let A be any preorder. We speak of A throughout this example as though it was the linear order given by time because this is a nice case, however the mathematics works for any $A \in \mathrm{Ob}(\mathbf{PrO})$.

There is a monad $\top = (T, \eta, \mu)$ that captures the idea that a function $f\colon X \to Y$ occurs in the context of time in the following sense: The output of f is determined not only by the element $x \in X$ on which it is applied but also by the time at which it was applied to x; and the output of f occurs at another time, which is not before the time of input.

The functor part of the monad is given on $X \in \mathrm{Ob}(\mathbf{Set})$ by

$$T(X) = \{p\colon A \to A \times X \mid \text{ if } p(a) = (a', x) \text{ then } a' \geqslant a\}.$$

The unit $\eta_X\colon X \to T(X)$ sends x to the function $a \mapsto (a, x)$. The multiplication map $\mu_X\colon T(T(X)) \to T(X)$ is roughly described as follows. If for every $a \in A$ you have a later element $a' \geqslant a$ and a function $p\colon A \to A \times X$ that takes elements of A to later elements of A and values of X, then $p(a')$ is a still later element of A and a value of X, as desired.

Morphisms in the Kleisli category $\mathbf{Kls}(\top)$ can be curried to be functions $f\colon A \times X \to A \times Y$ such that if $f(a,x) = (a', y)$ then $a' \geqslant a$.

Remark 5.3.3.9. One of the most important monads in computer science is the so-called *state monad*. It is used when one wants to allow a program to mutate state variables (e.g. in the program

if $x > 4$ then $x := x + 1$ else Print “done”)

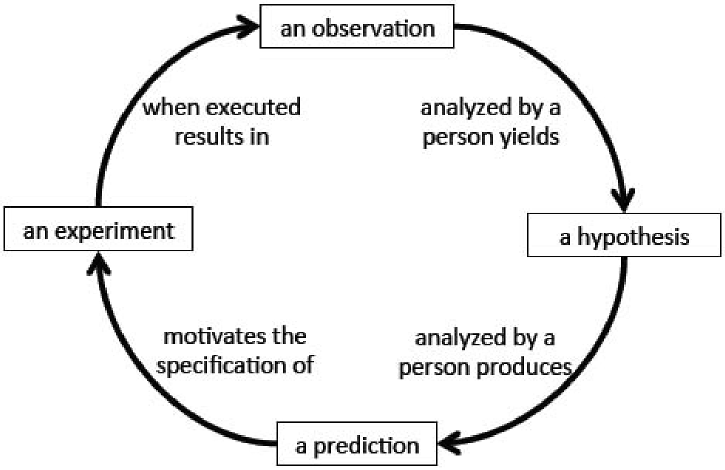
x is a state variable. The state monad is a special case of the monad discussed in Example 5.3.3.8. Given any set A, the usual *state monad of type* A is obtained by giving A the indiscrete preorder (see Example 3.4.4.5). More explicitly it is a monad with functor part

$$X \mapsto (A \times X)^X,$$

and it will be briefly discussed in Example 5.3.5.4.

Example 5.3.3.10. Here we reconsider the image from the front cover of this book, reproduced here.

[14] Actually, Definition 2.5.2.3 is about composing spans, but a relation $R \subseteq A \times B$ is a kind of span, $R \to A \times B$.


an observation
analyzed by a person yields
a hypothesis
analyzed by a person produces
a prediction
motivates the specification of
an experiment
when executed results in

It looks like an olog, and all ologs are database schemas (see Section 3.5.2.14). But how is “analyzed by a person yields” a function from observations to hypotheses? The very name belies the fact that it is an invalid aspect in the sense of Section 2.3.2.1, because given an observation there may be more than one hypothesis yielded, corresponding to which person is doing the observing. In fact, all of the arrows in this diagram correspond to some hidden context involving people: the prediction is dependent on who analyzes the hypothesis, the specification of an experiment is dependent on who is motivated to specify it, and experiments may result in different observations by different observers.

Without monads, the model of science proposed by this olog would be difficult to believe in. But by choosing a monad we can make explicit (and then hide from discourse) our implicit assumption that “of course this is all dependent on which human is doing the science”. The choice of monad is an additional modeling choice. Do we want to incorporate the partial order of time? Do we want the scientist to be modified by each function (i.e. the person is changed when analyzing an observation to yield a hypothesis)? These are all interesting possibilities.

One reasonable choice would be to use the state monad of type A, where A is the set of scientific models. This implies the following context: every morphism $f\colon X \to Y$ in the Kleisli category of this monad is really a morphism $f\colon X \times A \to Y \times A$; while ostensibly giving a map from X to Y, it is influenced by the scientific model under which it is performed, and its outcome yields a new scientific model.

Reading the olog in this context might look like this:

> A hypothesis (in the presence of a scientific model) analyzed by a person produces a prediction (in the presence of a scientific model), which motivates the specification of an experiment (in the presence of a scientific model), which when executed results in an observation (in the presence of a scientific model), which analyzed by a person yields a hypothesis (in the presence of a scientific model).

The parenthetical statements can be removed if we assume them to always be around, which can be done using the monad above.

5.3.3.11 Relaxing functionality constraint for ologs

In Section 2.3.2 we said that every arrow in an olog has to be English-readable as a sentence, and it has to correspond to a function. For example, the arrow

$$\boxed{\text{a person}}\xrightarrow{\text{has}}\boxed{\text{a child}} \tag{5.11}$$

comprises an readable sentence, but does not correspond to a function because a person may have no children or more than one child. We'll call olog in which every arrow corresponds to a function (the only option proposed so far in the book) a *functional olog*. Requiring that ologs be functional as we have been doing, comes with advantages and disadvantages. The main advantage is that creating a functional olog requires more conceptual clarity about the situation, and this has benefits for the olog-creator as well as for anyone to whom he or she tries to explain the situation. The main disadvantage is that creating a functional olog takes more time, and the olog takes up more space on the page.

In the context of the power set monad (see Exercise 5.3.2.3), a morphism $f\colon X\to Y$ between sets X and Y becomes a binary relation on X and Y, rather than a function, as seen in Exercise 5.3.3.5. So in that context, the arrow in (5.11) becomes valid. An olog in which arrows correspond to mere binary relations rather than functions might be called a *relational olog*.

5.3.4 Monads in databases

In this section we discuss how to record data in the presence of a monad. The idea is quite simple. Given a schema (category) $\mathcal{C}$, an ordinary instance is a functor $I\colon\mathcal{C}\to\mathbf{Set}$. But if $\top=(T,\eta,\mu)$ is a monad, then a *Kleisli* $\top$*-instance on* $\mathcal{C}$ is a functor $J\colon\mathcal{C}\to\mathbf{Kls}(\top)$. Such a functor associates to every object $c\in\mathrm{Ob}(\mathcal{C})$ a set $J(c)$, and to every arrow $f\colon c\to c'$ in $\mathcal{C}$ a morphism $J(f)\colon J(c)\to J(c')$ in $\mathbf{Kls}(\top)$. How does this look in terms of tables?

Recall that to represent an ordinary database instance $I\colon\mathcal{C}\to\mathbf{Set}$, we use a tabular format in which every object $c\in\mathrm{Ob}(\mathcal{C})$ is displayed as a table including one ID column and an additional column for every arrow emanating from c. In the ID column of table c were elements of the set $I(c)$ and in the column assigned to some arrow $f\colon c\to c'$ the cells were elements of the set $I(c')$.

To represent a *Kleisli* database instance $J\colon\mathcal{C}\to\mathbf{Kls}\top$ is similar; we again use a tabular format in which every object $c\in\mathrm{Ob}(\mathcal{C})$ is displayed as a table including one ID column and an additional column for every arrow emanating from c. In the ID column of table c are again elements of the set $J(c)$; however in the column assigned to some arrow $f\colon c\to c'$ are not elements of $J(c')$ but T-values in $J(c')$, i.e. elements of $T(J(c'))$.

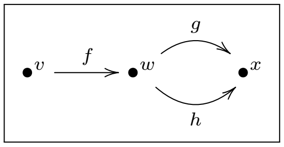
Example 5.3.4.1. Let $\top=(T,\eta,\mu)$ be the monad for partial functions, as discussed in Example 5.3.1.1. Given any schema $\mathcal{C}$, we can represent a Kleisli $\top$-instance $I\colon\mathcal{C}\to\mathbf{Kls}(\top)$ in tabular format. To every object $c\in\mathrm{Ob}(\mathcal{C})$ we'll have a set $I(c)$ of rows, and given a column $c\to c'$ every row will produce either a value in $I(c')$ or fail to produce a value; this is the essence of partial functions. We might denote the absence of a value using ☺.

Consider the schema indexing graphs

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

As we discussed in Section 4.2.1.20, an ordinary instance on $\mathcal{C}$ represents a graph.

$I :=$

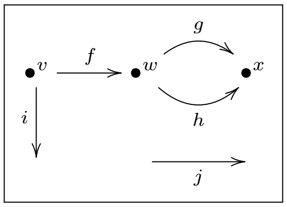


Arrow (I)		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x

Vertex (I)
ID
v
w
x

A Kleisli $\top$-instance on $\mathcal{C}$ represents graphs in which edges can fail to have a source vertex, fail to have a target vertex, or both.

$J :=$



Arrow (J)		
ID	**src**	**tgt**
f	v	w
g	w	x
h	w	x
i	v	☺
j	☺	☺

Vertex (J)
ID
v
w
x

The context of these tables is that of partial functions, so we do not need a reference for ☺ in the vertex table. Mathematically, the morphism $J(src)\colon J(\texttt{Arrow}) \to J(\texttt{Vertex})$ needs to be a function $J(\texttt{Arrow}) \to J(\texttt{Vertex}) \sqcup \{☺\}$, and it is.

5.3.4.2 Probability distributions

Let $[0,1] \subseteq \mathbb{R}$ denote the set of real numbers between 0 and 1. Let X be a set and $p\colon X \to [0,1]$ a function. We say that p is a *finitary probability distribution on* X if there exists a finite subset $W \subseteq X$ such that

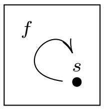
$$\sum_{w \in W} p(w) = 1, \tag{5.12}$$

and such that $p(x) > 0$ if and only if $x \in W$. Note that W is unique if it exists; we call it *the support of* p and denote it $\mathbf{Supp}(p)$. Note also that if X is a finite set then every function p satisfying (5.12) is a finitary probability distribution on X.

For any set X, let $\mathbf{Dist}(X)$ denote the set of finitary probability distributions on X. It is easy to check that given a function $f\colon X \to Y$ one obtains a function $\mathbf{Dist}(f)\colon \mathbf{Dist}(X) \to \mathbf{Dist}(Y)$ by $\mathbf{Dist}(f)(y) = \sum_{f(x)=y} p(x)$. Thus we can consider $\mathbf{Dist}\colon \mathbf{Set} \to \mathbf{Set}$ as a functor, and in fact the functor part of a monad. Its unit $\eta\colon X \to \mathbf{Dist}(X)$ is given by the Kronecker delta function $x \mapsto \delta_x$ where $\delta_x(x) = 1$ and $\delta_x(x') = 0$ for $x' \neq x$. Its multiplication $\mu\colon \mathbf{Dist}(\mathbf{Dist}(X)) \to \mathbf{Dist}(X)$ is given by weighted sum: given a finitary probability distribution $w\colon \mathbf{Dist}(X) \to [0,1]$ and $x \in X$, put $\mu(w)(x) = \sum_{p \in \mathbf{Supp}(w)} w(p)p(x)$.

Example 5.3.4.3 (Markov chains). Let $\mathcal{L}oop$ be the loop schema,

$\mathcal{L}oop :=$



as in Example 3.5.2.9. A **Dist**-instance on $\mathcal{L}oop$ is equivalent to a time-homogeneous Markov chain. To be explicit, a functor $\delta\colon \mathcal{L}oop \to \mathbf{KlsDist}$ assigns to the unique object

$s \in \mathrm{Ob}(\mathcal{L}oop)$ a set $S = \delta(s)$, which we call the state space, and to $f\colon s \to s$ a function $\delta(f)\colon S \to \mathbf{Dist}(S)$, which sends each element $x \in S$ to some probability distribution on elements of S. For example, the table δ on the left corresponds to the Markov matrix M on the right below:

$\delta :=$

s	
ID	**f**
1	.5(1)+.5(2)
2	1(2)
3	.7(1)+.3(3)
4	.4(1)+.3(2)+.3(4)

$$M := \begin{pmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.7 & 0 & 0.3 & 0 \\ 0.4 & 0.3 & 0 & 0.3 \end{pmatrix} \qquad (5.13)$$

As one might hope, for any natural number $n \in \mathbb{N}$ the map $f^n\colon S \to \mathbf{Dist}(S)$ corresponds to the matrix M^n, which sends an element in S to its probable location after n iterations of the transition map.

Application 5.3.4.4. Every star emits a spectrum of light, which can be understood as a distribution on the electromagnetic spectrum. Given an object B on earth, different parts of B will absorb radiation at different rates. Thus B produces a function from the electromagnetic spectrum to distributions of energy absorption. In the context of the probability distributions monad, we can record data on the schema

$$\begin{tikzcd}
\overset{\texttt{star}}{\bullet} \arrow[r, "\text{emits}"] & \overset{\texttt{wavelengths}}{\bullet} \arrow[r, "\text{absorbed by } B"] & \overset{\texttt{energies}}{\bullet}
\end{tikzcd}$$

The composition formula for Kleisli categories is the desired one: to each star we associate the weighted sum of energy absorption rates over the set of wavelengths emitted by the star.

◊◊

5.3.5 Monads and adjunctions

There is a strong connection between monads and adjunctions: every adjunction creates a monad, and every monad "comes from" an adjunction. For example, the List monad (Example 5.3.2.2) comes from the free-forgetful adjunction between sets and monoids

$$\begin{tikzcd}
\mathbf{Set} \arrow[r, shift left, "F"] & \mathbf{Mon} \arrow[l, shift left, "U"]
\end{tikzcd}$$

(see Proposition 5.1.1.2). That is, for any set X, the free monoid on X is

$$F(X) = (\mathrm{List}(X), [\,], +\!\!+),$$

and the underlying set of that monoid is $U(F(X)) = \mathrm{List}(X)$. Now it may seem like there was no reason to use monoids at all—the set $\mathrm{List}(X)$ was needed in order to discuss $F(X)$—but it will turn out that the unit η and multiplication μ will come drop out of the adjunction too. First, we discuss the unit and counit of an adjunction.

Definition 5.3.5.1. Let $\mathcal{C}$ and $\mathcal{D}$ be categories, and let $L\colon \mathcal{C} \to \mathcal{D}$ and $R\colon \mathcal{D} \to \mathcal{C}$ be functors with adjunction isomorphism

$$\alpha_{c,d}\colon \mathrm{Hom}_{\mathcal{D}}(L(c), d) \xrightarrow{\cong} \mathrm{Hom}_{\mathcal{C}}(c, R(d))$$

for any objects $c \in \mathrm{Ob}(\mathcal{C})$ and $d \in \mathrm{Ob}(\mathcal{D})$. The *unit* $\eta\colon \mathrm{id}_{\mathcal{C}} \to R \circ L$ (respectively the *counit* $\epsilon\colon L \circ R \to \mathrm{id}_{\mathcal{D}}$) are natural transformations defined as follows.

Given an object $c \in \mathrm{Ob}(\mathcal{C})$, we apply α to $\mathrm{id}_{L(c)}\colon L(c) \to L(c)$ to get

$$\eta_c\colon c \to R \circ L(c);$$

similarly given an object $d \in \mathrm{Ob}(\mathcal{D})$ we apply α^{-1} to $\mathrm{id}_{R(d)}\colon R(d) \to R(d)$ to get

$$\epsilon_d\colon L \circ R(d) \to d.$$

Below we will show how to use the unit and counit of any adjunction to make a monad. We first walk through the process in Example 5.3.5.2.

Example 5.3.5.2. Consider the adjunction $\mathbf{Set}\underset{U}{\overset{F}{\rightleftarrows}}\mathbf{Mon}$ between sets and monoids. Let $T = U \circ F\colon \mathbf{Set} \to \mathbf{Set}$; this will be the functor part of our monad, and we have $T = \mathrm{List}$. Then the unit of the adjunction, $\eta\colon \mathrm{id}_{\mathbf{Set}} \to U \circ F$ is precisely the unit of the monad: for any set $X \in \mathrm{Ob}(\mathbf{Set})$ the component $\eta_X\colon X \to \mathrm{List}(X)$ is the function that takes $x \in X$ to the singleton list $[x] \in \mathrm{List}(X)$. The monad also has a multiplication map $\mu_X\colon T(T(X)) \to T(X)$, which amounts to flattening a list of lists. This function comes about using the counit ϵ, as follows

$$T \circ T = U \circ F \circ U \circ F \xrightarrow{\mathrm{id}_U \diamond \epsilon \diamond \mathrm{id}_F} U \circ F = T.$$

The general procedure for extracting a monad from an adjunction is analogous to that shown in Example 5.3.5.2. Given any adjunction

$$\begin{tikzcd}\mathcal{C} \arrow[r, shift left, "L"] & \mathcal{D} \arrow[l, shift left, "R"]\end{tikzcd}$$

We define $\top = R \circ L\colon \mathcal{C} \to \mathcal{C}$, we define $\eta\colon \mathrm{id}_{\mathcal{C}} \to \top$ to be the unit of the adjunction (as in Definition 5.3.5.1), and we define $\mu\colon \top \circ \top \to \top$ to be the natural transformation $\mathrm{id}_R \diamond \epsilon \diamond \mathrm{id}_L\colon RLRL \to RL$, obtained by applying the counit $\epsilon\colon LR \to \mathrm{id}_{\mathcal{D}}$.

The above procedure produces monads on arbitrary categories $\mathcal{C}$, whereas our definition of monad (Definition 5.3.2.1) considers only the case $\mathcal{C} = \mathbf{Set}$. However, this definition can be generalized to arbitrary categories $\mathcal{C}$ by simply replacing every occurrence of the string **Set** with the string $\mathcal{C}$. Similarly, our definition of Kleisli categories (Definition 5.3.3.1) considers only the case $\mathcal{C} = \mathbf{Set}$, but again the generalization to arbitrary categories $\mathcal{C}$ is straightforward. In Proposition 5.3.5.3, it may be helpful to again put $\mathcal{C} = \mathbf{Set}$ if one is at all disoriented.

Proposition 5.3.5.3. *Let $\mathcal{C}$ be a category, let $(\top, \eta, \mu)$ be a monad on $\mathcal{C}$, and let $\mathcal{K} := \mathbf{Kls}_{\mathcal{C}}(\top)$ be the Kleisli category. Then there is an adjunction*

$$\begin{tikzcd}\mathcal{C} \arrow[r, shift left, "L"] & \mathcal{K} \arrow[l, shift left, "R"]\end{tikzcd}$$

such that the monad $(\top, \eta, \mu)$ is obtained (up to isomorphism) by the above procedure.

Sketch of proof. The functor $L\colon \mathcal{C} \to \mathcal{K}$ was discussed in Remark 5.3.3.3. We define it to be identity on objects (recall that $\mathrm{Ob}(\mathcal{K}) = \mathrm{Ob}(\mathcal{C})$). Given objects $c, c' \in \mathrm{Ob}(\mathcal{C})$ the function

$$\mathrm{Hom}_{\mathcal{C}}(c, c') \xrightarrow{L} \mathrm{Hom}_{\mathcal{K}}(c, c') = \mathrm{Hom}_{\mathcal{C}}(c, \top(c'))$$

is given by $f \mapsto \eta_{c'} \circ f$. The fact that this is a functor (i.e. that it preserves composition) follows from a monad axiom.

The functor $R\colon \mathcal{K} \to \mathcal{C}$ acts on objects by sending $c \in \mathrm{Ob}(\mathcal{K}) = \mathrm{Ob}(\mathcal{C})$ to $\top(c) \in \mathrm{Ob}(\mathcal{C})$. For objects $c, c' \in \mathrm{Ob}(\mathcal{K})$ the function

$$\mathrm{Hom}_{\mathcal{C}}(c, \top(c')) = \mathrm{Hom}_{\mathcal{K}}(c, c') \xrightarrow{R} \mathrm{Hom}_{\mathcal{C}}(\top(c), \top(c'))$$

is given by sending the $\mathcal{C}$-morphism $f\colon c \to \top(c')$ to the composite

$$\top(c) \xrightarrow{\top(f)} \top\top(c') \xrightarrow{\mu_{c'}} \top(c').$$

Again, the functoriality follows from monad axioms.

We will not continue on to show that these are adjoint or that they produce the monad $(\top, \eta, \mu)$, but see [Mac, VI.5.1] for the remainder of the proof.

□

Example 5.3.5.4. Let $A \in \mathrm{Ob}(\mathbf{Set})$ be a set, and recall the currying adjunction

$$\begin{tikzcd} \mathbf{Set} \arrow[r, shift left, "A\times -"] & \mathbf{Set} \arrow[l, shift left, "-^A"] \end{tikzcd}$$

discussed briefly in Example 5.1.1.8. The corresponding monad St_A is typically called the *state monad of type* A in programming language theory. Given a set X, we have

$$St_A(X) = (A \times X)^A.$$

In the Kleisli category $\mathbf{Kls}(St_A)$ a morphism from X to Y is a function of the form $X \to (A \times Y)^A$, but this can be curried to a function $A \times X \to A \times Y$.

This monad is related to holding on to an internal state variable of type A. Every morphism ostensibly from X to Y actually takes as input not only an element of X but also the current state $a \in A$, and it produces as output not only an element of Y but an updated state as well.

Computer scientists in programming language theory have found monads to be very useful ([Mog]). In much the same way, monads on **Set** can be useful in databases, as discussed in Section 5.3.4. Another, totally different way to use monads in databases is by using a mapping between schemas to produce in each one an internal model of the other. That is, for any functor $F\colon \mathcal{C} \to \mathcal{D}$, i.e. mapping of database schemas, the adjunction (Σ_F, Δ_F) produces a monad on $\mathcal{C}$–**Set**, and the adjunction (Δ_F, Π_F) produces a monad on $\mathcal{D}$–**Set**. If one interprets the List monad as producing in **Set** an internal model of the category **Mon** of monoids, one can similarly interpret the above monads on $\mathcal{C}$–**Set** and $\mathcal{D}$–**Set** as producing internal models of each within the other.

5.4 Operads

In this section we briefly introduce operads, which are generalizations of categories. They often are useful for speaking about self-similarity of structure. For example, we will use them to model agents made up of smaller agents, or materials made up of smaller materials. This association with self-similarity is not really inherent in the definition, but it tends to emerge in our thinking about many operads used in practice.

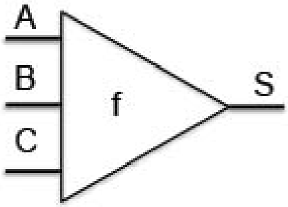
Let me begin with a warning.

Warning 5.4.0.5. My use of the term operad is not entirely standard and conflicts with widespread usage. The more common term for what I am calling an operad is *symmetric colored operad* or a *symmetric multicategory*. An operad classically is a multicategory with one object, and a colored operad is a multicategory. The analogy is that "operad is to multicategory as monoid is to category". The term multicategory stems from the fact that the morphisms in a multicategory have many, rather than one, input. But there is nothing really "multi" about the multicategory itself, only its morphisms. Probably the real reason though is that I find the term multicategory to be clunky and the term operad to be sleek, clocking in at half the syllables. I apologize if my break with standard terminology causes any confusion.

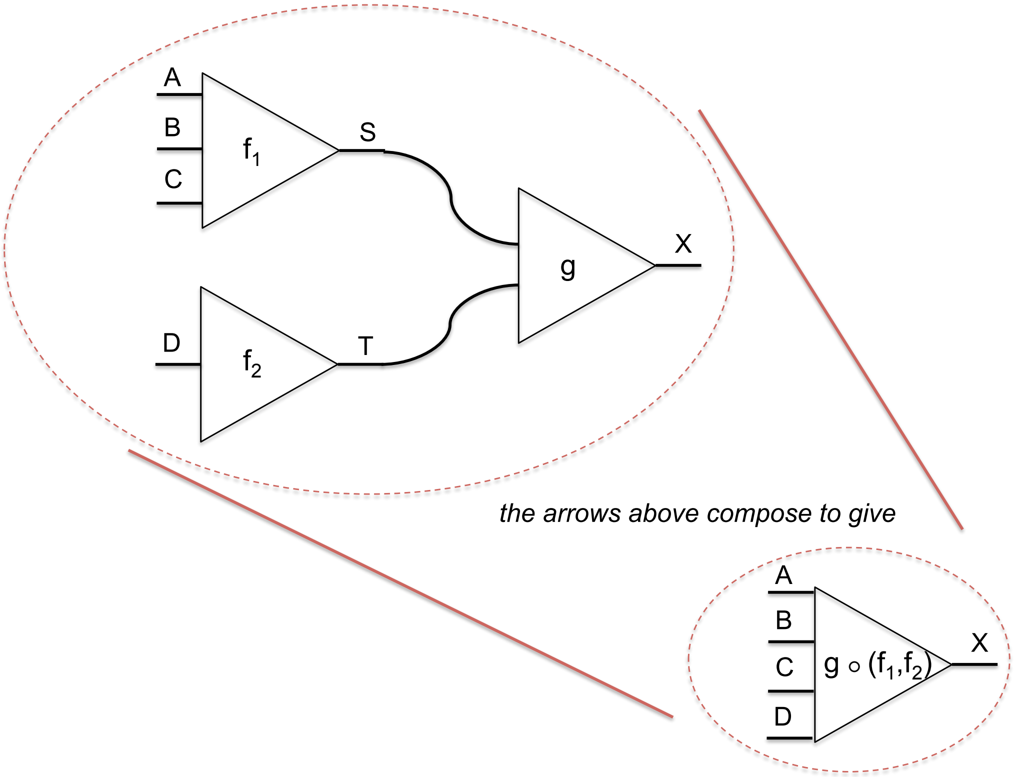
This introduction to operads is quite short. One should see [Le1] for an excellent treatment.

5.4.1 Definition and classical examples

An operad is like a category in that it has objects, morphisms, and a composition formula, and it follows an identity law and an associativity law. The difference is that each morphism has many inputs (and one output).



The description of composition in an operad is a bit heavier than it is in a category, but the idea fairly straightforward. Here is a picture of morphisms being composed.



Note that S and T disappear from the composition, but this is analogous to the way the middle object disappears from the composition of morphisms in a category

$$\boxed{A \xrightarrow{f} S \xrightarrow{g} X} \qquad \textit{the arrows to the left compose to give} \qquad \boxed{A \xrightarrow{g\circ f} X}$$

Here is the definition, which we take directly from [Sp4].

Definition 5.4.1.1. An *operad* $\mathcal{O}$ is defined as follows: One announces some constituents (A. objects, B. morphisms, C. identities, D. compositions) and asserts that they conform to some laws (1. identity law, 2. associativity law). Specifically,

A. one announces a collection $\text{Ob}(\mathcal{O})$, each element of which is called an *object* of $\mathcal{O}$.

B. for each object $y \in \text{Ob}(\mathcal{O})$, finite set $n \in \text{Ob}(\mathbf{Fin})$, and n-indexed set of objects $x\colon n \to \text{Ob}(\mathcal{O})$, one announces a set $\mathcal{O}_n(x; y) \in \text{Ob}(\mathbf{Set})$. Its elements are called *morphisms from x to y* in $\mathcal{O}$.

C. for every object $x \in \text{Ob}(\mathcal{O})$, one announces a specified morphism denoted $\text{id}_x \in \mathcal{O}_1(x; x)$ called *the identity morphism on x.*

D. Let $s\colon m \to n$ be a morphism in $\mathbf{Fin}$. Let $z \in \text{Ob}(\mathcal{O})$ be an object, let $y\colon n \to \text{Ob}(\mathcal{O})$ be an n-indexed set of objects, and let $x\colon m \to \text{Ob}(\mathcal{O})$ be an m-indexed set of objects. For each element $i \in n$, write $m_i := s^{-1}(i)$ for the pre-image of s under i, and write $x_i = x|_{m_i}\colon m_i \to \text{Ob}(\mathcal{O})$ for the restriction of x to m_i. Then one announces a function

$$\circ\colon \mathcal{O}_n(y;z)\times\prod_{i\in n}\mathcal{O}_{m_i}(x_i;y(i))\longrightarrow\mathcal{O}_m(x;z), \tag{5.14}$$

called *the composition formula.*

Given an n-indexed set of objects $x\colon n\to\mathrm{Ob}(\mathcal{O})$ and an object $y\in\mathrm{Ob}(\mathcal{O})$, we sometimes abuse notation and denote the set of morphisms from x to y by $\mathcal{O}(x_1,\ldots,x_n;y)$. [15] We may write $\mathrm{Hom}_{\mathcal{O}}(x_1,\ldots,x_n;y)$, in place of $\mathcal{O}(x_1,\ldots,x_n;y)$, when convenient. We can denote a morphism $\phi\in\mathcal{O}_n(x;y)$ by $\phi\colon x\to y$ or by $\phi\colon(x_1,\ldots,x_n)\to y$; we say that each x_i is a *domain object* of ϕ and that y is the *codomain object* of ϕ. We use infix notation for the composition formula, e.g. writing $\psi\circ(\phi_1,\ldots,\phi_n)$.

One asserts that the following laws hold:

1. for every $x_1,\ldots,x_n,y\in\mathrm{Ob}(\mathcal{O})$ and every morphism $\phi\colon(x_1,\ldots,x_n)\to y$, we have
$$\phi\circ(\mathrm{id}_{x_1},\ldots,\mathrm{id}_{x_n})=\phi \qquad\text{and}\qquad \mathrm{id}_y\circ\phi=\phi;$$

2. Let $m\xrightarrow{s}n\xrightarrow{t}p$ be composable morphisms in **Fin**. Let $z\in\mathrm{Ob}(\mathcal{O})$ be an object, let $y\colon p\to\mathrm{Ob}(\mathcal{O})$, $x\colon n\to\mathrm{Ob}(\mathcal{O})$, and $w\colon m\to\mathrm{Ob}(\mathcal{O})$ respectively be a p-indexed, n-indexed, and m-indexed set of objects. For each $i\in p$, write $n_i=t^{-1}(i)$ for the pre-image and $x_i\colon n_i\to\mathrm{Ob}(\mathcal{O})$ for the restriction. Similarly, for each $k\in n$ write $m_k=s^{-1}(k)$ and $w_k\colon m_k\to\mathrm{Ob}(\mathcal{O})$; for each $i\in p$, write $m_{i,-}=(t\circ s)^{-1}(i)$ and $w_{i,-}\colon m_{i,-}\to\mathrm{Ob}(\mathcal{O})$; for each $j\in n_i$, write $m_{i,j}:=s^{-1}(j)$ and $w_{i,j}\colon m_{i,j}\to\mathrm{Ob}(\mathcal{O})$. Then the diagram below commutes:

$$\begin{tikzcd}
& \mathcal{O}_p(y;z)\times\prod_{i\in p}\mathcal{O}_{n_i}(x_i;y(i))\times\prod_{i\in p,\ j\in n_i}\mathcal{O}_{m_{i,j}}(w_{i,j};x_i(j)) \arrow[dl] \arrow[dr] & \\
\mathcal{O}_n(x;z)\times\prod_{k\in n}\mathcal{O}_{m_k}(w_k;x(k)) \arrow[dr] & & \mathcal{O}_p(y;z)\times\prod_{i\in p}\mathcal{O}_{m_{i,-}}(w_{i,-};y(i)) \arrow[dl] \\
& \mathcal{O}_m(w;z) &
\end{tikzcd}$$

Remark 5.4.1.2. In this remark we will discuss the abuse of notation in Definition 5.4.1.1 and how it relates to an action of a symmetric group on each morphism set in our definition of operad. We follow the notation of Definition 5.4.1.1, especially following the use of subscripts in the composition formula.

Suppose that $\mathcal{O}$ is an operad, $z\in\mathrm{Ob}(\mathcal{O})$ is an object, $y\colon n\to\mathrm{Ob}(\mathcal{O})$ is an n-indexed set of objects, and $\phi\colon y\to z$ is a morphism. If we linearly order n, enabling us to write $\phi\colon(y(1),\ldots,y(|n|))\to z$, then changing the linear ordering amounts to finding an isomorphism of finite sets $\sigma\colon m\xrightarrow{\cong}n$, where $|m|=|n|$. Let $x=y\circ\sigma$ and for each $i\in n$, note that $m_i=\sigma^{-1}(\{i\})=\{\sigma^{-1}(i)\}$, so $x_i=x|_{\sigma^{-1}(i)}=y(i)$. Taking $\mathrm{id}_{x_i}\in\mathcal{O}_{m_i}(x_i;y(i))$ for each $i\in n$, and using the identity law, we find that the composition formula induces a bijection $\mathcal{O}_n(y;z)\xrightarrow{\cong}\mathcal{O}_m(x;z)$, which we might denote by

$$\sigma\colon\mathcal{O}(y(1),y(2),\ldots,y(n);z)\cong\mathcal{O}\big(y(\sigma(1)),y(\sigma(2)),\ldots,y(\sigma(n));z\big).$$

[15] There are three abuses of notation when writing $\mathcal{O}(x_1,\ldots,x_n;y)$, which we will fix one by one. First, it confuses the set $n\in\mathrm{Ob}(\mathbf{Fin})$ with its cardinality $|n|\in\mathbb{N}$. But rather than writing $\mathcal{O}(x_1,\ldots,x_{|n|};y)$, it would be more consistent to write $\mathcal{O}(x(1),\ldots,x(|n|);y)$, because we have assigned subscripts another meaning in part D. But even this notation unfoundedly suggests that the set n has been endowed with a linear ordering, which it has not. This may be seen as a more serious abuse, but see Remark 5.4.1.2.

In other words, there is an induced group action of $\mathrm{Aut}(n)$ on $\mathcal{O}_n(x; z)$, where $\mathrm{Aut}(n)$ is the group of permutations of an n-element set.

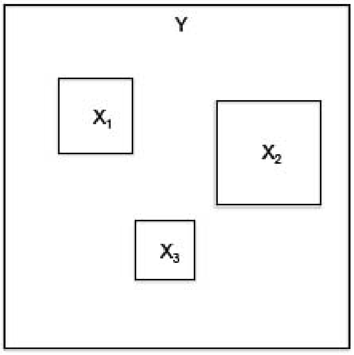
Throughout this book, we will permit ourselves to abuse notation and speak of morphisms $\phi\colon (x_1, x_2, \ldots, x_n) \to y$ for a natural number $n \in \mathbb{N}$, without mentioning the abuse inherent in choosing an order, so long as it is clear that permuting the order of indices would not change anything up to canonical isomorphism.

Example 5.4.1.3. Let **Sets** denote the operad defined as follows. For objects we put $\mathrm{Ob}(\mathbf{Sets}) = \mathrm{Ob}(\mathbf{Set})$. For a natural number $n \in \mathbb{N}$ and sets $X_1, \ldots, X_n, Y$, put

$$\mathrm{Hom}_{\mathbf{Sets}}(X_1, \ldots, X_n; Y) := \mathrm{Hom}_{\mathbf{Set}}(X_1 \times \cdots \times X_n, Y).$$

Given functions $f_1\colon (X_{1,1} \times \cdots \times X_{1,m_1}) \to Y_1$ through $f_n\colon (X_{n,1} \times \cdots \times X_{n,m_n}) \to Y_n$ and a function $Y_1 \times \cdots \times Y_n \to Z$, the universal property provides us a unique function of the form $(X_{1,1} \times \cdots \times X_{n,m_n}) \longrightarrow Z$, giving rise to our composition formula.

Example 5.4.1.4 (Little squares operad). An operad commonly used in mathematics is called the *little n-cubes operad*. We'll focus on $n = 2$ and talk about the little squares operad $\mathcal{O}$. Here the set of objects has only one element, which we denote by a square, $\mathrm{Ob}(\mathcal{O}) = \{\square\}$. For a natural number $n \in \mathbb{N}$, a morphism $f\colon (\square, \square, \ldots, \square) \longrightarrow \square$ is a positioning of n non-overlapping squares inside of a square. Here is a picture of a morphism $(X_1, X_2, X_3) \to Y$, where $X_1 = X_2 = X_3 = Y = \square$.



The composition law says that given a positioning of small squares inside a large square, and given a positioning of tiny squares inside each of those small squares, we get a positioning of tiny squares inside a large square. A picture is shown in Figure 5.15.

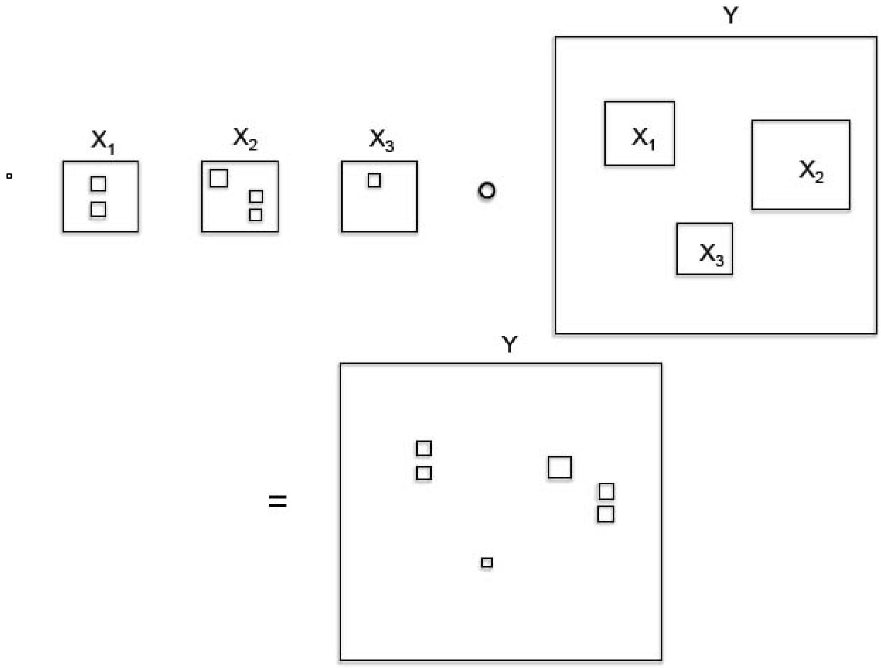


Figure 5.15: Here we show a morphism $(X_1, X_2, X_3) \to Y$ and morphisms $(W_{1,1}, W_{1,2}) \to X_1$, $(W_{2,1}, W_{2,2}, W_{2,3}) \to X_2$, and $(W_{3,1}) \to X_3$, each of which is a positioning of squares inside a square. The composition law scales and positions the squares in the "obvious" way.

Hopefully, what we meant by "self-similarity" in the introduction to this section (see page 247) is becoming clear.

Exercise 5.4.1.5. Consider an operad $\mathcal{O}$ like the little squares operad from Example 5.4.1.4, except with three objects: square, circle, equilateral triangle. A morphism is again a non-overlapping positioning of shapes inside of a shape.

a.) Draw an example of a morphism f from two circles and a square to a triangle.

b.) Find three other morphisms that compose into f, and draw the composite.

◊

5.4.1.6 Operads: functors and algebras

If operads are like categories, then we can define things like functors and call them *operad functors*. Before giving the definition, we give a warning.

Warning 5.4.1.7. What we call operad functors in Definition 5.4.1.8 are usually (if not always) called *operad morphisms*. We thought that the terminology clash between morphisms *of* operads and morphisms *in* an operad was too confusing. It is similar to what would occur in regular category theory (e.g. Chapter 4) if we replaced the term "functor" with the term "category morphism".

Definition 5.4.1.8. Let $\mathcal{O}$ and $\mathcal{O}'$ be operads. An *operad functor from* $\mathcal{O}$ *to* $\mathcal{O}'$, denoted $F\colon \mathcal{O} \to \mathcal{O}'$ consists of some constituents (A. on-objects part, B. on-morphisms part) conforming to some laws (1. preservation of identities, 2. preservation of composition), as follows:

A. There is a function $\mathrm{Ob}(F)\colon \mathrm{Ob}(\mathcal{O}) \to \mathrm{Ob}(\mathcal{O}')$.

B. For each object $y \in \mathrm{Ob}(\mathcal{O})$, finite set $n \in \mathrm{Ob}(\mathbf{Fin})$, and n-indexed set of objects $x\colon n \to \mathrm{Ob}(\mathcal{O})$, there is a function

$$F_n\colon \mathcal{O}_n(x;y) \to \mathcal{O}'_n(Fx;Fy).$$

As in B. above, we often denote $\mathrm{Ob}(F)$, and also each F_n, simply by F. The laws that govern these constituents are as follows:

1. For each object $x \in \mathrm{Ob}(\mathcal{O})$, the equation $F(\mathrm{id}_x) = \mathrm{id}_{Fx}$ holds.
2. Let $s\colon m \to n$ be a morphism in **Fin**. Let $z \in \mathrm{Ob}(\mathcal{O})$ be an object, let $y\colon n \to \mathrm{Ob}(\mathcal{O})$ be an n-indexed set of objects, and let $x\colon m \to \mathrm{Ob}(\mathcal{O})$ be an m-indexed set of objects. Then, with notation as in Definition 5.4.1.1, the following diagram of sets commutes:

$$\begin{tikzcd}
\mathcal{O}_n(y;z) \times \prod_{i\in n} \mathcal{O}_{m_i}(x_i;y(i)) \arrow[r, "F"] \arrow[d, "\circ"'] & \mathcal{O}'_n(Fy;Fz) \times \prod_{i\in n} \mathcal{O}'_{m_i}(Fx_i;Fy(i)) \arrow[d, "\circ"] \\
\mathcal{O}_m(x;z) \arrow[r, "F"'] & \mathcal{O}'_m(Fx;Fz)
\end{tikzcd} \tag{5.16}$$

We denote the category of operads and operad functors by **Oprd**.

Exercise 5.4.1.9. Let $\mathcal{O}$ denote the little squares operad from Example 5.4.1.4 and let $\mathcal{O}'$ denote the operad you constructed in Exercise 5.4.1.5.

a.) Can you come up with an operad functor $\mathcal{O} \to \mathcal{O}'$?

b.) Is it possible to find an operad functor $\mathcal{O}' \to \mathcal{O}$?

◊

Definition 5.4.1.10 (Operad algebra)**.** Let $\mathcal{O}$ be an operad. An *algebra on* $\mathcal{O}$ is an operad functor $A\colon \mathcal{O} \to \mathbf{Sets}$.

Remark 5.4.1.11. Every category can be construed as an operad (yes, there is a functor **Cat** → **Oprd**), by simply not including non-unary morphisms. That is, given a category $\mathcal{C}$, one makes an operad $\mathcal{O}$ with $\mathrm{Ob}(\mathcal{O}) := \mathrm{Ob}(\mathcal{C})$ and with

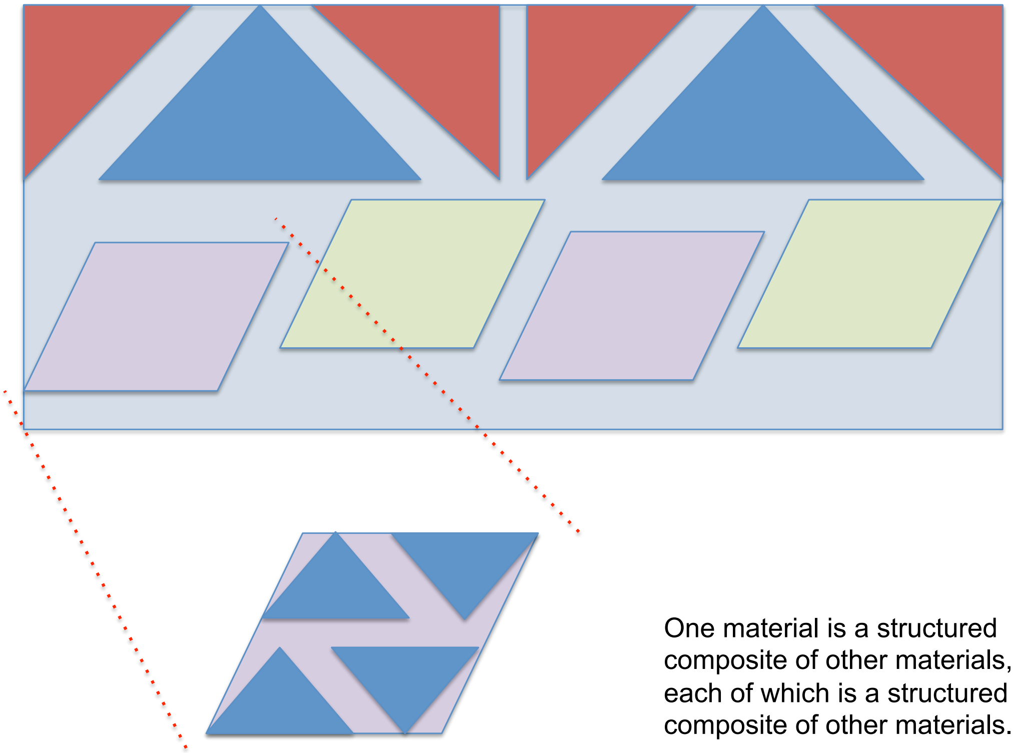
$$\mathrm{Hom}_{\mathcal{O}}(x_1,\ldots,x_n;y) = \begin{cases} \mathrm{Hom}_{\mathcal{C}}(x_1,y) & \text{if } n = 1; \\ \varnothing & \text{if } n \neq 1 \end{cases}$$

Just like a schema is a category presentation, it is possible to discuss operad presentations by generators and relations. Under this analogy, an algebra on an operad corresponds to an instance on a schema.

5.4.2 Applications of operads and their algebras

Hierarchical structures may be well-modeled by operads. Describing such structures using operads and their algebras allows one to make appropriate distinctions between different types of thinking. For example, the allowable formations are encoded in the operad, whereas the elements that will fit into those formations are encoded in the algebra. Morphisms of algebras are high-level understandings of how elements of very different types (such as materials vs. numbers) can occupy the same place in the structure and be compared. We will give examples below.

Application 5.4.2.1. Every material is composed of constituent materials, arranged in certain patterns. (In case the material is "pure", we consider the material to consist of itself as the sole constituent.) Each of these constituent materials each is itself an arrangement of constituent materials. Thus we see a kind of self-similarity which we can model with operads.



One material is a structured composite of other materials, each of which is a structured composite of other materials.

(5.17)

For example, a tendon is made of collagen fibers that are assembled in series and then in parallel, in a specific way. Each collagen fibre is made of collagen fibrils that are again assembled in series and then in parallel, with slightly different specifications. We can continue down, perhaps indefinitely, though our resolution fails at some point. A collagen fibril is made up of tropocollagen collagen molecules, which are twisted ropes of collagen molecules, etc.[16]

Here is how operads might be employed. We want the same operad to model both actual materials, theoretical materials, and functional properties; that is we want more than one algebra on the same operad.

The operad $\mathcal{O}$ should abstractly model the structure, but not the substance being structured. Imagine that each of the shapes (including the background "shape") in Diagram (5.17) is a place-holder, saying something like "*your material here*". Each morphism (that's what (5.17) is a picture of) represents a construction of a material out of parts. In our picture, it appears we are only concerned with the spacial arrangements, but there is far more flexibility than that. Whether we want to allow for additional details beyond spacial arrangements is the kinds of choice we make in a meeting called "what operad should we use?"

◊◊

Application 5.4.2.2. Suppose we have chosen an operad $\mathcal{O}$ to model the structure of materials. Each object of $\mathcal{O}$ might correspond to a certain quality of material, and each morphism corresponds to an arrangement of various qualities to form a new quality. An algebra $A\colon \mathcal{O} \to \mathbf{Sets}$ on $\mathcal{O}$ forces us to choose what substances will fill in for these qualities. For every object $x \in \mathrm{Ob}(\mathcal{O})$, we want a set $A(x)$ which will be the set of materials with that quality. For every arrangement, i.e. morphism, $f\colon (x_1, \ldots, x_n) \to y$, and every choice $a_1 \in A(x_1), \ldots, a_n \in A(x_n)$ of materials, we need to understand what material $a' = A(f)(a_1, \ldots, a_n) \in A(y)$ will emerge when these materials are arranged in accordance with f. We are really pinning ourselves down here.

But there may be more than one interesting algebra on $\mathcal{O}$. Suppose that $B\colon \mathcal{O} \to \mathbf{Sets}$ is an algebra of strengths rather than materials. For each object $x \in \mathrm{Ob}(\mathcal{O})$, which represents some quality, we let $B(x)$ be the set of possible strengths that something of quality x can have. Then for each arrangement, i.e. morphism, $f\colon (x_1, \ldots, x_n) \to y$, and every choice $b_1 \in B(x_1), \ldots, b_n \in B(x_n)$ of strengths, we need to understand what strength $b' = B(f)(b_1, \ldots, b_n) \in B(y)$ will emerge when these strengths are arranged in accordance with f. Certainly an impressive achievement!

Finally, a morphism of algebras $S\colon A \to B$ would consist of a coherent system for assigning to each material $a \in A(X)$ of a given quality x a specific strength $S(a) \in B(X)$, in such a way that morphisms behaved appropriately. In this language we have stated a very precise goal for the field of material mechanics.

◊◊

Exercise 5.4.2.3. Consider again the little squares operad $\mathcal{O}$ from Example 5.4.1.4. Suppose we wanted to use this operad to describe those photographic mosaics.

a.) Come up with an algebra $P\colon \mathcal{O} \to \mathbf{Sets}$ that sends the square to the set of all photos that can be pasted into that square. What does P do on morphisms in $\mathcal{O}$?

b.) Come up with an algebra $C\colon \mathcal{O} \to \mathbf{Sets}$ that sends each square to the set of all colors (visible frequencies of light). In other words, $C(\square)$ is the set of colors, not the set of ways to color the square. What does C do on morphisms in $\mathcal{O}$. Hint: use some kind of averaging scheme for the morphisms.

c.) Guess: if someone were to appropriately define morphisms of $\mathcal{O}$-algebras (something akin to natural transformations between functors $\mathcal{O} \to \mathbf{Sets}$), do you think there

[16] Thanks to Professor Sandra Shefelbine for explaining the hierarchical nature of collagen to me. Any errors are my own.

would some a morphism of algebras $P \to C$?

◊

5.4.2.4 Wiring diagrams

Example 5.4.2.5. Here we describe an *operad of relations*, which we will denote by $\mathcal{R}$. The objects are sets, $\mathrm{Ob}(\mathcal{R}) = \mathrm{Ob}(\mathbf{Set})$. A morphism $f\colon (x_1, x_2, \ldots, x_n) \longrightarrow x'$ in $\mathcal{R}$ is a diagram in **Set** of the form

$$\begin{tikzcd}
& & & R \arrow[dlll, "f_1"'] \arrow[dll, "f_2"] \arrow[dl, phantom, "\cdots"] \arrow[d, "f_n"] \arrow[dr, "f'"] & \\
x_1 & x_2 & \cdots & x_n & x'
\end{tikzcd} \tag{5.18}$$

such that the induced function $R \longrightarrow (x_1 \times x_2 \times \cdots \times x_n \times x')$ is an injection.

We use a composition formula similar to that in Definition 2.5.2.3. Namely, we form a fiber product

$$\begin{tikzcd}
& & FP \arrow[dl] \arrow[dr] & & \\
& \prod_{i\in\underline{n}} R_i \arrow[dl] \arrow[dr] & & S \arrow[dl] \arrow[dr] & \\
\prod_{i\in\underline{n}} \prod_{j\in\underline{m_i}} x_{i,j} & & \prod_{i\in\underline{n}} y_i & & z
\end{tikzcd}$$

One can show that the induces function $FP \longrightarrow \left(\prod_{i\in\underline{n}} \prod_{j\in\underline{m_i}} x_i\right) \times y$ is an injection, so we have a valid composition formula. Finally, the associativity and identity laws hold.[17]

Application 5.4.2.6. Suppose we are trying to model life in the following way. We define an entity as a set of phenomena, but in order to use colloquial language we say the entity *is able to experience* that set of phenomena. We also want to be able to put entities together to form a super-entity, so we have a notion of morphism $f\colon (e_1, \ldots, e_n) \longrightarrow e'$ defined as a relation as in (5.18). The idea is that the morphism f is a way of translating between the phenomena that may be experienced by the sub-entities and the phenomena that may be experienced by the super-entity.

The operad $\mathcal{R}$ from Example 5.4.2.5 becomes useful as a language for discussing issues in this domain. ◊◊

Example 5.4.2.7. Let $\mathcal{R}$ be the operad of relations from Example 5.4.2.5. Consider the algebra $S\colon \mathcal{R} \to \mathbf{Sets}$ given by $S(x) = \mathbb{P}(x)$. Given a morphism $\prod_i x_i \leftarrow R \to y$ and subsets $x'_i \subseteq x_i$, we have a subset $\prod_i x'_i \subseteq \prod_i x_i$. We take the fiber product

$$\begin{tikzcd}
& FP \arrow[rr] \arrow[dl] & & R \arrow[dl] \arrow[dr] & \\
\prod_i x'_i \arrow[rr] & & \prod_i x_i & & y
\end{tikzcd}$$

[17] Technically we need to use isomorphism classes of cone points, but we don't worry about this here.

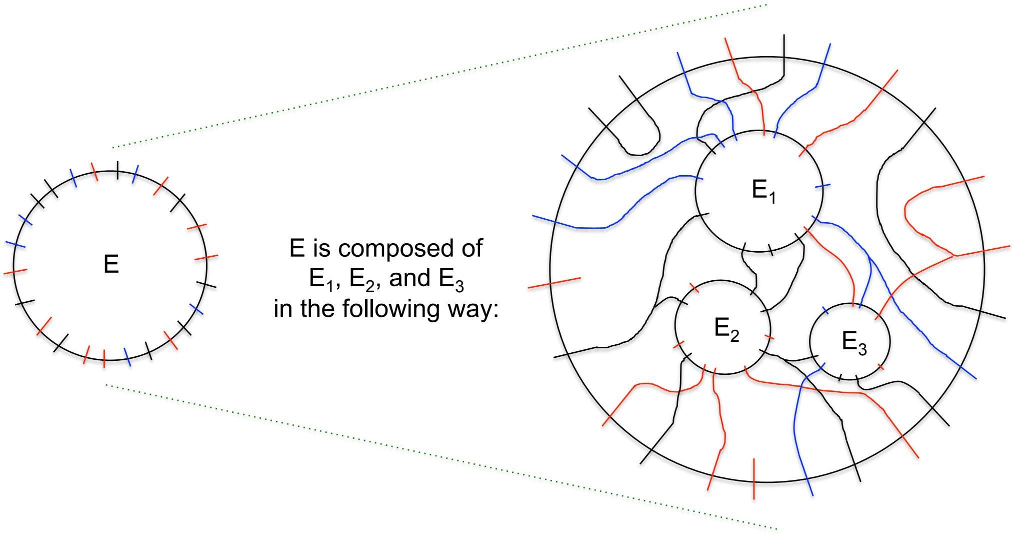
and the image of $FP \to y$ is a subset of y.

Application 5.4.2.8. Following Application 5.4.2.6 we can use Example 5.4.2.7 as a model of survival. Each entity survives only for a subset of the phenomena that it can experience. Under this interpretation, the algebra from Example 5.4.2.7 defines survival as the survival of all parts. That is, suppose that we understand how a super-entity is composed of sub-entities in the sense that we have a translation between the set of phenomena that may be experienced across the sub-entities and the set of phenomena that may be experienced by the super-entity. Then the super-entity will survive exactly those phenomena which translate to phenomena for which each sub-entity desires.

Perhaps a better term than survival would be "allowance". A bureaucracy consists of a set of smaller bureaucracies, each of which allows certain phenomena to pass; the whole bureaucracy allows something to pass if and only if, when translated to the perspective of each sub-bureaucracy, it is allowed to pass there.

◊◊

Example 5.4.2.9. In this example we discuss wiring diagrams that look like this:



The operad in question will be denoted $\mathcal{W}$; it is discussed in greater detail in [Sp4]. The objects of $\mathcal{W}$ are pairs (C, s) where C is a finite set and $v\colon C \to \mathrm{Ob}(\mathbf{Set})$ is a function. Think of such an object as a circle with C-many cables sticking out of it; each cable c is assigned a set $v(c)$ corresponding to the set of values that can be carried on that cable. For example $E_2 = (C, v)$ where $|C| = 11$ and we consider v to be specified by declaring that black wires carry $\mathbb{Z}$ and red wires carry {sweet, sour, salty, bitter, umami}.

The morphisms in $\mathcal{W}$ will be pictures as above, formalized as follows. Given objects $(C_1, v_1), \ldots, (C_n, v_n), (D, w)$, a morphism $F\colon ((C_1, v_1), \ldots, (C_n, v_n)) \longrightarrow (D, w)$ is a commutative diagram of sets [18]

$$\begin{tikzcd}
\bigsqcup_{i\in\underline{n}} C_i \arrow[r, "i"] \arrow[dr, "\sqcup_i v_i"'] & G \arrow[d, "x"] & D \arrow[l, "j"'] \arrow[dl, "w"] \\
 & \mathrm{Ob}(\mathbf{Set}) &
\end{tikzcd}$$

such that i and j are jointly surjective.

Composition of morphisms is easily understood in pictures: given wiring diagrams inside of wiring diagrams, we can throw away the intermediary circles. In terms of sets, we perform a pushout.

There is an operad functor $\mathcal{W}\to\mathcal{S}$ given by sending (C,v) to $\prod_{c\in C} v(c)$. The idea is that to an entity defined as having a bunch of cables carrying variables, a phenomenon is the same thing as a choice of value on each cable. A wiring diagram translates between values experienced locally and values experienced globally.

Application 5.4.2.10. In cognitive neuroscience or in industrial economics, it may be that we want to understand the behavior of an entity such as a mind, a society, or a business in terms of its structure. Knowing the connection pattern (connectome, supply chain) of sub-entities should help us understand how big changes are generated from small ones.

Under the functor $\mathcal{W}\to\mathcal{S}$ the algebra $\mathcal{S}\to\mathbf{Sets}$ from Application 5.4.2.8 becomes an algebra $\mathcal{W}\to\mathbf{Sets}$. To each entity we now associate some subset of the value-assignments it can carry. ◊◊

Application 5.4.2.11. In [RS], Radul and Sussman discuss propagator networks. These can presumably be understood in terms of wiring diagrams and their algebra of relations. ◊◊

[18] If one is concerned with cardinality issues, fix a cardinality κ and replace $\mathrm{Ob}(\mathbf{Set})$ everywhere with $\mathrm{Ob}(\mathbf{Set}_{<\kappa})$.

Index

Bibliography

[Ati] Atiyah, M. (1989) “Topological quantum field theories”. *Publications Mathématiques de l’IHÉS* 68 (68), pp. 175–186.

[Axl] Axler, S. (1997) *Linear algebra done right.* Springer.

[Awo] S. Awodey. (2010) *Category theory.* Second edition. Oxford Logic Guides, 52. Oxford University Press, Oxford.

[Bar] Bralow, H. (1961) “Possible principles underlying the transformation of sensory messages”. *Sensory communication*, pp. 217 – 234.

[BD] Baez, J.C.; Dolan, J. (1995) “Higher-dimensional algebra and topological quantum field theory”. *Journal of mathematical physics* vol 36, 6073.

[BFL] Baez, J.C.; Fritz, T.; Leinster, T. (2011) “A characterization of entropy in terms of information loss.” *Entropy* 13, no. 11.

[BS] Baez, J.C.; Stay, M. (2011) “Physics, topology, logic and computation: a Rosetta Stone.” *New structures for physics*, 95Ð172. Lecture Notes in Phys., 813, Springer, Heidelberg.

[BP1] Brown, R.; Porter, T. (2006) “Category Theory: an abstract setting for analogy and comparison, In: *What is Category Theory?* Advanced Studies in Mathematics and Logic, Polimetrica Publisher, Italy, pp. 257-274.

[BP2] Brown, R.; Porter, T. (2003) “Category theory and higher dimensional algebra: potential descriptive tools in neuroscience”, *Proceedings of the International Conference on Theoretical Neurobiology, Delhi*, edited by Nandini Singh, National Brain Research Centre, Conference Proceedings 1 80-92.

[BW] M. Barr, C. Wells. (1990) *Category theory for computing science.* Prentice Hall International Series in Computer Science. Prentice Hall International, New York.

[Big] Biggs, N.M. (2004) *Discrete mathematics.* Oxford University Press, NY.

[Dia] Diaconescu, R. (2008) *Institution-independent model theory* Springer.

[DI] Döring, A.; Isham, C. J. “A topos foundation for theories of physics. I. Formal languages for physics.” *J. Math. Phys.* 49 (2008), no. 5, 053515.

[EV] Ehresmann, A.C.; Vanbremeersch, J.P. (2007) *Memory evolutive systems; hierarchy, emergence, cognition*. Elsevier.

[Eve] Everett III, H. (1973). "The theory of the universal wave function." In *The many-worlds interpretation of quantum mechanics* (Vol. 1, p. 3).

[Gog] Goguen, J. (1992) "Sheaf semantics for concurrent interacting objects" *Mathematical structures in Computer Science* Vol 2, pp. 159 – 191.

[Gro] Grothendieck, A. (1971). *Séminaire de Géométrie Algébrique du Bois Marie - 1960-61 - Revêtements étales et groupe fondamental - (SGA 1)* (Lecture notes in mathematics 224) (in French). Berlin; New York: Springer-Verlag.

[Kro] Krömer, R. (2007). *Tool and Object: A History and Philosophy of Category Theory*, Birkhauser.

[Lam] Lambek, J. (1980) "From λ-calculus to Cartesian closed categories". In *Formalism*, Academic Press, London, pp. 375 – 402.

[Law] Lawvere, F.W. (2005) "An elementary theory of the category of sets (long version) with commentary." (Reprinted and expanded from Proc. Nat. Acad. Sci. U.S.A. **52** (1964)) *Repr. Theory Appl. Categ.* 11, pp. 1 – 35.

[Kho] Khovanov, M. (2000) "A categorificiation of the Jones polynomial" *Duke Math J.*.

[Le1] Leinster, T. (2004) *Higher Operads, Higher Categories.* London Mathematical Society Lecture Note Series 298, Cambridge University Press.

[Le2] Leinster, T. (2012) "Rethinking set theory". ePrint available http://arxiv.org/abs/1212.6543.

[Lin] Linsker, R. (1988) "Self-organization in a perceptual network". *Computer* 21, no. 3, pp. 105 – 117.

[LM] Landry, E.; Marquis, J-P., 2005, "Categories in Contexts: historical, foundational, and philosophical." *Philosophia Mathematica*, (3), vol. 13, no. 1, 1 – 43.

[LS] F.W. Lawvere, S.H. Schanuel. (2009) *Conceptual mathematics. A first introduction to categories.* Second edition. Cambridge University Press, Cambridge.

[MacK] MacKay, D.J. (2003). *Information theory, inference and learning algorithms.* Cambridge university press.

[Mac] Mac Lane, S. (1998) *Categories for the working mathematician.* Second edition. Graduate Texts in Mathematics, 5. Springer-Verlag, New York.

[Mar1] Marquis, J-P. (2009) *From a Geometrical Point of View: a study in the history and philosophy of category theory*, Springer.

[Mar2] Marquis, J-P, "Category Theory", *The Stanford Encyclopedia of Philosophy* (Spring 2011 Edition), Edward N. Zalta (ed.), http://plato.stanford.edu/archives/spr2011/entries/category-theory

[Min] Minsky, M. *The Society of Mind.* Simon and Schuster, NY 1985.

[Mog] Moggi, E. (1989) “A category-theoretic account of program modules.” *Category theory and computer science (Manchester, 1989),* 101Ð117, Lecture Notes in Comput. Sci., 389, Springer, Berlin.

[nLa] nLab authors. http://ncatlab.org/nlab/show/HomePage

[Pen] Penrose, R. (2006) *The road to reality.* Random house.

[RS] Radul, A.; Sussman, G.J. (2009). “The art of the propagator”. *MIT Computer science and artificial intelligence laboratory technical report.*

[Sp1] Spivak, D.I. (2012) “Functorial data migration”. *Information and communication*

[Sp2] Spivak, D.I. (2012) “Queries and constraints via lifting problems”. Submitted to *Mathematical structures in computer science.* ePrint available: http://arxiv.org/abs/1202.2591

[Sp3] Spivak, D.I. (2012) “Kleisli database instances”. ePrint available: http://arxiv.org/abs/1209.1011

[Sp4] Spivak, D.I. (2013) “The operad of wiring diagrams: Formalizing a graphical language for databases, recursion, and plug-and-play circuits”. Available online: http://arxiv.org/abs/1305.0297

[SGWB] Spivak D.I., Giesa T., Wood E., Buehler M.J. (2011) “Category Theoretic Analysis of Hierarchical Protein Materials and Social Networks.” PLoS ONE 6(9): e23911. doi:10.1371/journal.pone.0023911

[SK] Spivak, D.I., Kent, R.E. (2012) “Ologs: A Categorical Framework for Knowledge Representation.” *PLoS ONE* 7(1): e24274. doi:10.1371/journal.pone.0024274.

[WeS] Weinberger, S. (2011) “What is... Persistent Homology?” AMS.

[WeA] Weinstein, A. (1996) “Groupoids: unifying internal and external symmetry. *Notices of the AMS* Vol 43, no. 7, pp. 744 – 752.

[Wik] Wikipedia (multiple authors). Various articles, all linked with a hyperreference are scattered throughout this text. All accessed December 6, 2012 – September 17, 2013.

18.S996 Category Theory for Scientists
Spring 2013