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 $m_1 m_2 m_3 m_4 m_5$ 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 $\{\smiley\}$ denote a one-element set.

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

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

◊

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 $\{(x m_i y \sim x m_i' y) \mid x, y \in \mathrm{List}(G), 1 \leqslant i \leqslant n\}$, 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,

$$[b,c,b,d,d,a,c,a,a,c,d] = [b,c,b,a,a,c,a,c,a,a,c,d] = [b,c,b,a,a,a,c,d] \\ = [b,c,b,a,d,d,d].$$

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}.)

◊◊

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 $[a,b,c,d,e,f] = [a,b,c]$. ◊

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?

◊

3.1.1.23 Cyclic monoids

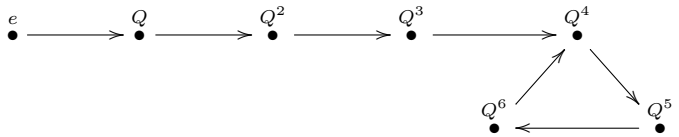
Definition 3.1.1.24. A monoid is called *cyclic* if it has a presentation involving only one generator.

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. ◊

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

$$\curvearrowright : M \times S \to S$$

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

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

Remark 3.1.2.2. To be pedantic (and because it's sometimes useful), we may rewrite $\curvearrowright$ 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 $\curvearrowright\colon \mathbb{N} \times S \to S$ by taking a pair (n, s) to the remainder that appears when $n + s$ is divided by 12. For example $4 \curvearrowright 2 = 6$ and $8 \curvearrowright 9 = 5$. 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

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

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 $x \cdot - \colon T \to T$, 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 $\mathrm{List}(B)$ on B.

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

b.) Can you think of a state s of G, and two distinct elements $\ell, \ell' \in \mathrm{List}(B)$ 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 $\ell \in \mathrm{List}(B)$, one has $\ell \circlearrowright s = s$?

◊

Application 3.1.2.6. Let $f \colon \mathbb{R} \to \mathbb{R}$ 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.

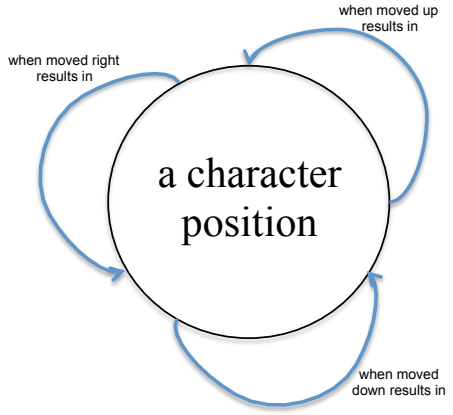
3.1.2.8 Monoids actions as ologs

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.

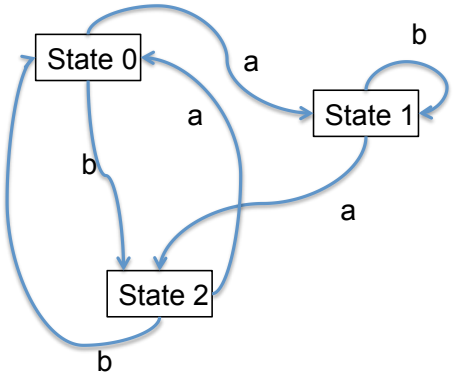


Figure 3.1: A finite state machine with alphabet $\Sigma = \{a, b\}$ and state set $S = \{\text{State 0, State 1, State 2}\}$. If pressed, we will make State 0 the initial state and {State 2} the set of final states.

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* $\mathrm{List}(\Sigma)$ *on S.*

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

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

We need to prove that there is an isomorphism of sets

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

Given an element $\epsilon\colon \mathrm{List}(\Sigma) \times S \to S$ 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 $\psi\colon \mathrm{Hom}_{\mathbf{Set}}(\Sigma \times S, S) \to A$.

Given an element $f \in \mathrm{Hom}_{\mathbf{Set}}(\Sigma \times S, S)$, define $\psi(f)\colon \mathrm{List}(\Sigma) \times S \to S$ on a pair $(L, s) \in \mathrm{List}(\Sigma) \times S$, 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 $\mathrm{List}(\Sigma)$ 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 (row, col), where $row \in S$ and $col \in G$, we put the element $\alpha(col, row) \in S$.

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							
$\mathbb{N}$	**0**	**2**	**3**	**5**	**7**	**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 $\cdot : \mathbb{N} \times S \to S$ 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 $f : \mathcal{M} \to \mathcal{M}'$, is a function $f : M \to M'$ 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 $i : \mathbb{N} \to \mathbb{Z}$ 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 $i_5 : \mathbb{N} \to \mathbb{Z}$ 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 $f\colon\mathbb{N}\to\mathrm{List}(\Sigma)$ given by sending $1\mapsto[a,b,b]$. The monoid action $\alpha\colon\mathrm{List}(\Sigma)\times S\to S$ 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.

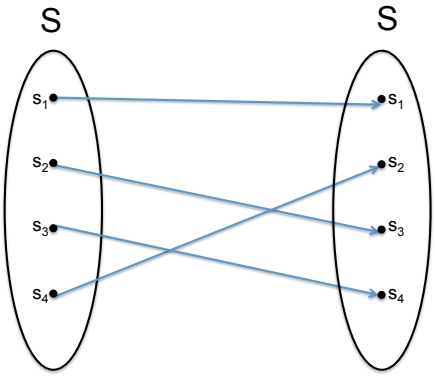
Application 3.2.1.6. In crystallography one is often concerned with the symmetries that arise in the arrangement A of atoms in a molecule. To think about symmetries in terms of groups, we first define an *atom-arrangement* to be a finite subset $i\colon A \subseteq \mathbb{R}^3$. A symmetry in this case 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:

$$\begin{array}{ccc} A & \dashrightarrow & A \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle i} \\ \mathbb{R}^3 & \xrightarrow[f]{} & \mathbb{R}^3 \end{array}$$

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 $\curvearrowright\colon G \times S \to S$ such that for all $s \in S$ and $g, g' \in G$, we have

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

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 $\theta \curvearrowright (x, y, z) = (x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta, z)$, or with matrix notation as

$$\theta \curvearrowright (x, y, z) := (x, y, z) \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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 $\mathbb{R}^3$			
$\mathbb{R}^3$	$\theta = 45^\circ$	$\theta = 90^\circ$	$\theta = 100^\circ$
(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 $\theta \curvearrowright (x, y, z) \in S$; 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 $U(1) \curvearrowright S$ extends to an action $U(1) \curvearrowright A$. 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 $\curvearrowright\colon G \times X \to X$. 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

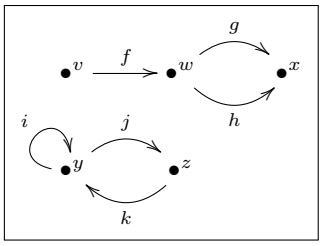
Definition 3.3.1.1. A *graph* G consists of a sequence $G := (V, A, src, tgt)$ where

- 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.

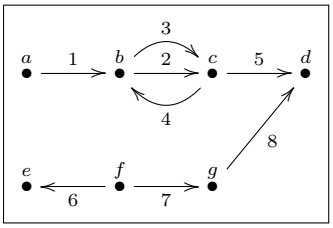
Example 3.3.1.3. Every olog has an underlying graph. The additional information in an olog has to do with which pairs of paths are declared equivalent, as well as text that has certain English-readability rules.

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 $(A \sqcup B, A, src, tgt)$. ◊

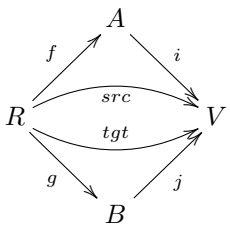
Exercise 3.3.1.6.

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?

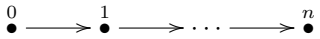
◊

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.



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.

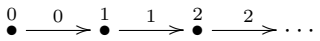
Example 3.3.1.8. Let $n \in \mathbb{N}$ be a natural number. The *chain graph of length* n, denoted $[n]$ is the graph depicted here:



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

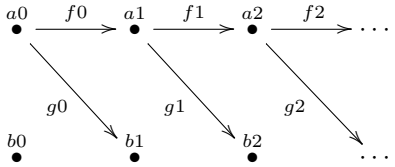


When we get to limits in a category, we will understand that products can be taken in the category of graphs (see Example 4.5.1.5), and $N \times G$ will make sense. For now, we construct it by hand.

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.

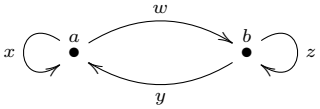
Let G be the graph drawn below

Then $T(G)$ will be the graph



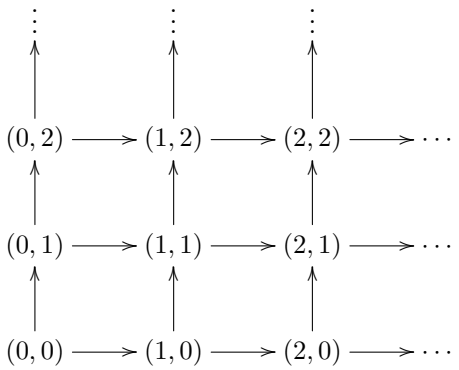
As you can see, f-arrows still take a's to a's and g-arrows still take a's to b's, but they always march forward in time.

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 $p \in \mathrm{Path}_G^{(n)}$ is a head-to-tail sequence

$$p = (v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \xrightarrow{a_3} \dots \xrightarrow{a_n} v_n) \tag{3.5}$$

of arrows in G, which we denote by $v_0 a_1 a_2 \dots a_n$. In particular we have canonical isomorphisms $\mathrm{Path}_G^{(1)} \cong A$ and $\mathrm{Path}_G^{(0)} \cong V$; 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,

$$\mathrm{Path}_G := \bigcup_{n \in \mathbb{N}} \mathrm{Path}_G^{(n)}.$$

Every path $p \in \mathrm{Path}_G$ has a source vertex and a target vertex, and we may denote these by $\overline{src}, \overline{tgt} \colon \mathrm{Path}_G \to V$. 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 $p\colon v \to w$.

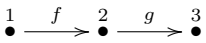
There is a concatenation operation on paths. Given a path $p\colon v \to w$ and $q\colon w \to x$, we define the concatenation, denoted $pq\colon v \to x$ in the obvious way. If $p = va_1, a_2 \dots a_m$ and $q = wb_1b_2 \dots b_n$ then $pq = va_1 \dots a_m b_1 \dots b_n$. In particular, if p (resp. 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

$$\{yi^{p_1}(jk)^{q_1} \cdots i^{p_n}(jk)^{q_n} \mid n, p_1, q_1, \dots, p_n, q_n \in \mathbb{N}\}$$

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? ◊

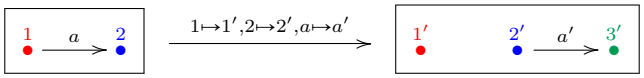
3.3.3 Graph homomorphisms

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:

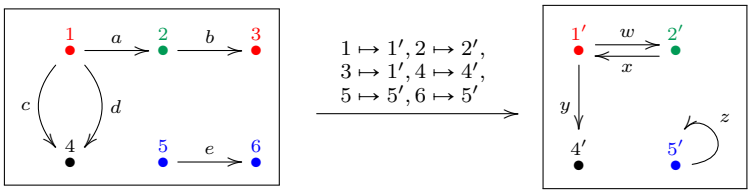
$$\begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ {\scriptstyle src}\downarrow & & \downarrow{\scriptstyle src'} \\ V & \xrightarrow[f_0]{} & V' \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ {\scriptstyle tgt}\downarrow & & \downarrow{\scriptstyle tgt'} \\ V & \xrightarrow[f_0]{} & V' \end{array} \tag{3.6}$$

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 $f_0 : V \to V'$. Given that choice, condition (3.6) imposes in this case that there is a unique choice of graph homomorphism $f : G \to G'$.

Exercise 3.3.3.4.

a.) Where are a, b, c, d, e sent under $f_1 : A \to A'$ 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 $f : G \to G'$, there an induced function $\mathrm{Path}(f) : \mathrm{Path}(G) \to \mathrm{Path}(G')$.

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 $i : A \to V \times V$ be function guaranteed by the universal property for products, as applied to $src, tgt : A \to V$. One might hope to summarize Condition (3.6) for graph homomorphisms by the commutativity of the single square

$$\begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle i'} \\ V \times V & \xrightarrow[f_0 \times f_0]{} & V' \times V'. \end{array} \tag{3.8}$$

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
⋮	⋮

$n = 5m$	
m	n
0	0
1	5
2	10
3	15
4	20
5	25
6	30
⋮	⋮

$\lvert n - m \rvert \leqslant 1$	
m	n
0	0
0	1
1	0
1	1
1	2
2	1
2	2
⋮	⋮

(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 $(S, \leqslant)$.

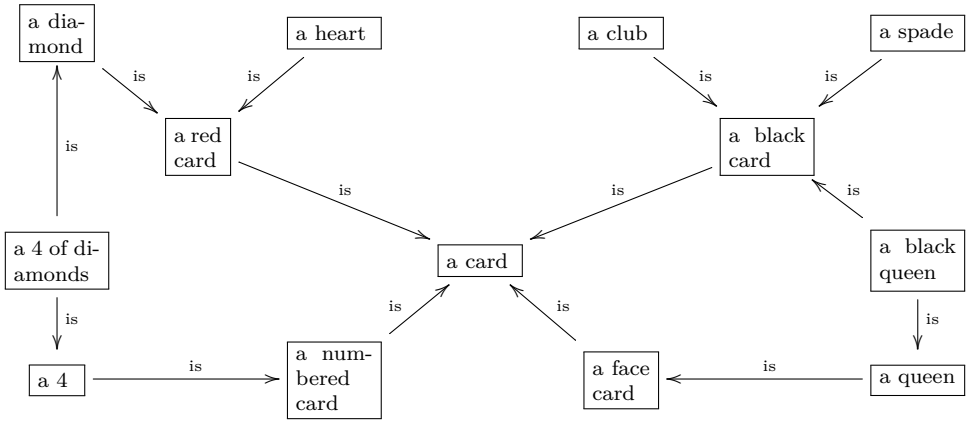
Exercise 3.4.1.2.

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,

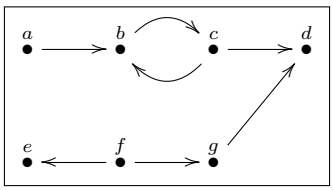
4. for any two vertices a, b there is an arrow $a \to b$ or an arrow $b \to a$.

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, $s \vee t = s \cup t$.

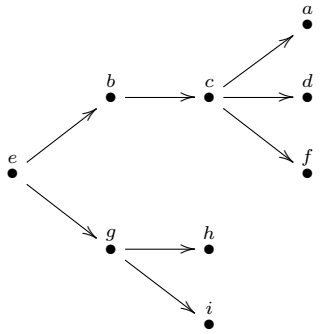
a.) Is it possible to put a monoid structure on the set S in which the multiplication formula is given by meets? If so, what would the identity element be?

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 $\mathcal{S}^{\mathrm{op}}$ is the preorder $(S, \leqslant^{\mathrm{op}})$ having the same set of elements but where $s \leqslant^{\mathrm{op}} s'$ 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 $\mathcal{N}^{\mathrm{op}}$ 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 $\mathcal{S}^{\mathrm{op}}$ also a partial order?

b.) If $\mathcal{S}$ is a linear order, is $\mathcal{S}^{\mathrm{op}}$ 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 $\mathcal{S}^{\mathrm{op}}$ has the same elements as $\mathcal{S}$. Is t the join of s_1 and s_2 in $\mathcal{S}^{\mathrm{op}}$, 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 $(M, \leqslant_C) \longrightarrow (M, \leqslant_S^{\mathrm{op}})$?"

◊◊

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?

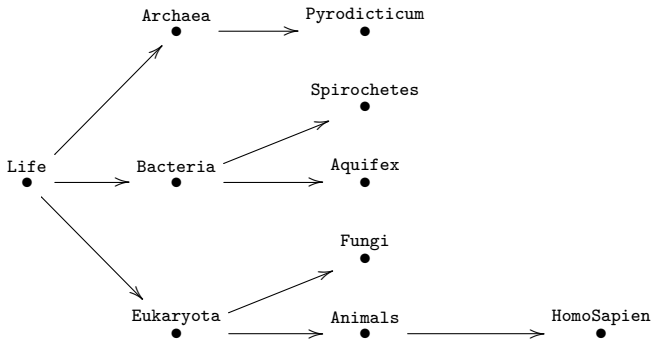
◊

3.4.5 Other applications

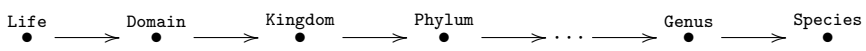
3.4.5.1 Biological classification

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 $f : C \to [n]$, 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)

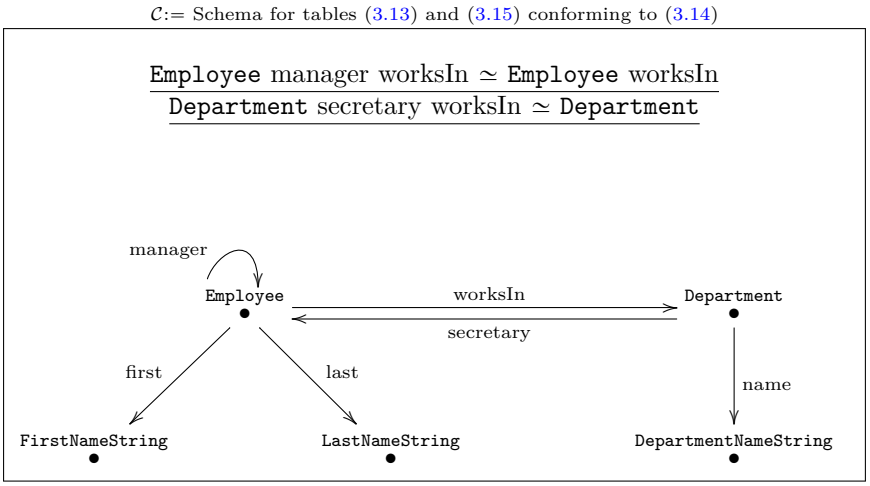
In combination, Displays (3.13) and (3.15) form a collection of tables with the property that every column is either a primary key or a foreign key. The notion of data column is now subsumed under the notion of foreign key column. Everything is either a primary key (one per table, labeled ID) or a foreign 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:



(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?

$$\texttt{Employee}\ \text{worksIn secretary} \overset{?}{\simeq} \texttt{Employee}$$

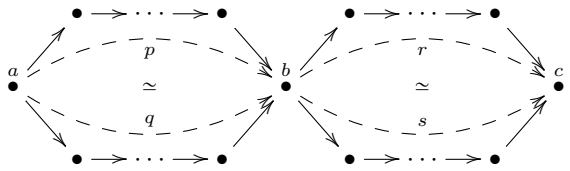
c.) What about this one?

$$\texttt{Department}\ \text{secretary manager worksIn name} \overset{?}{\simeq} \texttt{Department}\ \text{name}$$

◊

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.

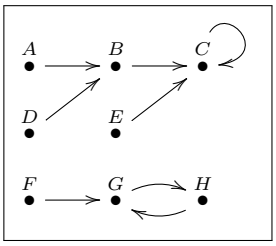
Example 3.5.2.9 (Discrete dynamical system). Consider the schema

$$\mathcal{L}oop := \quad \boxed{\begin{array}{c} f \\ \circlearrowright \\ s \end{array}} \tag{3.17}$$

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...

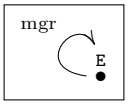


(3.18)

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

$$\texttt{E mgr mgr mgr mgr mgr mgr mgr mgr} \simeq \texttt{E mgr mgr mgr mgr mgr mgr mgr}$$

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?

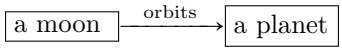
b.) How do the rules for ending the game in a win or draw play out in this model? (Look up online how chess games end if you don't know.)

◊

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



(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.

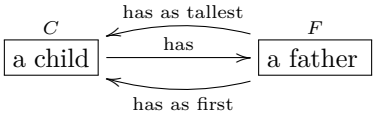
a.) Write down a schema, in terms of the boxes ⌜a thing I own⌝ and ⌜a place⌝ and one additional arrow, that might help one remember where they decided to put "random" things.

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



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?

◊

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 $(\mathrm{PK}, \mathrm{FK})\colon \mathcal{C} \to \mathbf{Set}$, 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 $\mathrm{PK}\colon V \to \mathbf{Set}$; i.e. to each vertex $v \in V$ one provides a set $\mathrm{PK}(v)$;[10] and

B. for every arrow $a \in A$ with $v = src(a)$ and $w = tgt(a)$, a function $\mathrm{FK}(a)\colon \mathrm{PK}(v) \to \mathrm{PK}(w)$. [11]

One asserts that the following law holds for any vertices v, w and paths $p = va_1a_2 \dots a_m$ and $q = va'_1a'_2 \dots a'_n$ from v to w:

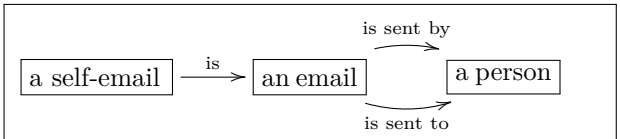
1. If $p \simeq q$ then for all $x \in \mathrm{PK}(v)$, we have

$$\mathrm{FK}(a_m) \circ \cdots \circ \mathrm{FK}(a_2) \circ \mathrm{FK}(a_1)(x) = \mathrm{FK}(a'_n) \circ \cdots \circ \mathrm{FK}(a'_2) \circ \mathrm{FK}(a'_1)(x)$$

in $\mathrm{PK}(w)$.

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 $\mathrm{PK}(v)$ 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 $r \in \mathrm{PK}(v)$ the (r, a) cell will contain the datum $\mathrm{FK}(a)(r)$.

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 PK(⌜an email⌝)?

b.) What is the set PK(⌜a person⌝)?

c.) What is the function FK(is sent by): PK(⌜an email⌝) → PK(⌜a person⌝)?

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?

◊

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. ◊

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? ◊

3.5.3.6 Paths through a database

Let $\mathcal{C} := (G, \simeq)$ be a schema and let $(\mathrm{PK}, \mathrm{FK})\colon \mathcal{C} \to \mathbf{Set}$ be an instance on $\mathcal{C}$. Then for every arrow $a\colon v \to w$ in G we get a function $\mathrm{FK}(a)\colon \mathrm{PK}(v) \to \mathrm{PK}(w)$. 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

$$\mathrm{FK}(p) := \mathrm{FK}(a_n) \circ \cdots \mathrm{FK}(a_2) \circ \mathrm{FK}(a_1)\colon \mathrm{PK}(v_0) \to \mathrm{PK}(v_n),$$

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 $\mathrm{PK}(s) = \{A, B, C, D, E, F, G, H\}$, interpret p as a function $\mathrm{PK}(s) \to \mathrm{PK}(s)$, 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 $(\mathrm{PK}, \mathrm{FK})$ 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?

◊