Basic Concepts in Mathematics

JT Paasch

Rough Draft - Last compiled August 3, 2021

CONTENTS

The goal of this book is to introduce many of the basic concepts in modern math. The book is divided into different parts, each of which covers a separate topic.

The first four parts (on Mathematical Method, Sets, Functions, and Relations) form the foundation that all of the other parts depend on. Any of the further parts beyond that foundation can be read independently.

Part [1]

THE MATHEMATICAL METHOD

How do mathematicians think? How do mathematicians investigate problems? How do mathematicians discover new knowledge? If a mathematician believes that they have figured out something new, how do they determine if they are correct?

In Part 1, we introduce the **mathematical method** by discussing how it is that mathematicians actually think and do their work.

[1]

DEDUCTIVE REASONING

PURE MATH IS A DISCIPLINE done through pure thinking. In this respect, math is a lot like philosophy. Both philosophers and mathematicians can do the entirety of their career's work, while sitting in an armchair and just *thinking*.

Of course, a mathematician might have some scratch paper beside them, to scribble things down while they think (and a big blackboard is really just a big piece of scratch paper). And naturally, a mathematician might want to write down and publish the conclusions they come to. But in principle, the real work consists in the thinking part.

Ponder. What do *you* think math is? What do you think the **proper subject** of study is for a mathematician? Where do you think you got those ideas?

So what exactly is it that mathematicians think about? If you are anything like me, you may have grown up with the impression that math is all about *calculation*. Perhaps you got this idea from school (I did). For a long time, I think I thought that the job of a mathematician was to sit there, and calculate

bigger and bigger numbers. (And I guess if you manage to calculate the biggest number so far, you win the game.)

But math is not really like this. Not at all. Of course, there can be some calculation involved, on occasion, when it is helpful. But calculation is really a part of *applied math*. When it comes to *pure math*, which is the topic we are concerned with here, the mathematician is typically not interested merely in calculating some particular number. On the contrary, the mathematician is usually much more interested in figuring out facts about a whole *class* of numbers, or figuring out what a number even *is*.

Ponder. How would you define a **number**? Can you give me a definition of a number without using the word "number" (or an equivalent concept) in your definition?

Moreover, a great deal of math doesn't even deal with numbers. More fundamentally, mathematicians spend their time dealing with *abstract structures* of various kinds. So, in what follows, let us to think about math as the **abstract study of structures**. The mathematician is interested in understanding in a deep way what these structures are like, and what kinds of properties they have.

Terminology. An **abstract structure** is a structure that is lifted away from irrelevant concrete details. For instance, instead of looking at a system of roads that connect cities, we can look at it as points, connected by lines.

1.1 What is Deductive Reasoning?

HOW DO WE STUDY ABSTRACT STRUCTURES? How do we learn things about abstract structures? Mathematicians employ a very distinctive method to come up with the conclusions that they come up with. This method is sometimes referred to as *deductive reasoning*.

Essentially, it is a logic game. Before we start the game, we agree to accept some basic *starting points*. Then, during the game, our goal is to derive new conclusions — conclusions that follow logically from our starting points. The primary rule of the game is this: you cannot use anything but cold, hard logic to figure out the new conclusions.

Remark 1.1. Think of the **starting points** as a small number of statements that we all agree to accept as "given" during the course of our investigation. Why do we need this? Well, we need to start somewhere, so in order to get our investigations going, we agree on some particular starting points.

During play, we are *strict* about the rules. We insist that we cannot use anything other than the information contained in our agreed upon starting points, and pure logic to move forward. So no new information is permitted, once we have started. Hence, the *only* things we can conclude, are things that **strictly follow** (logically) from our starting points.

That is what deductive reasoning is all about. **Deductive reasoning** is a process of reasoning where we start from some agreed upon starting points, and we use nothing but pure logic to figure what follows. We call it "deductive" because we use rigorous logical thinking to *deduce* the conclusions from our given starting points.

You might be thinking of Sherlock Holmes at this point, and you would be right to do so. Sherlock Holmes is one of fiction's most famous practitioners of deductive reasoning. He would always start with a set of facts gathered as evidence (these are his starting points), and invariably he would then use pure logic to figure out further facts that follow logically from his given starting points. This is the heart and soul of deductive reasoning.

1.2 An Example

TO GET A SENSE of how the deductive process works, let's consider a made-up example, which is not taken from the realm of math. Suppose we live in a small town which has a quaint public park at its center called "Hamfordton Park." This nice little park has the usual amenities. It has a playground for the children, it has a large grass field for playing sports, it has a few sidewalks meandering around the perimeter, and there is a selection of benches resting under shady trees.

Suppose that after an unfortunate accident, the city council established a statute which reads as follows:

> **Statute 2.34**:
> Vehicles are prohibited in Hamfordton Park.

For our purposes here, let's reword this a little bit, so that it has the shape of an "if-then" statement, like this:

> **Statute 2.34*** (Statute 2.34 reworded):
> *If* a vehicle is in Hamfordton Park, *then* it is in Hamfordton park illegally.

Remark 1.2. It is okay to reword statements, so long as we **preserve the meaning** of the original statement. English (or any other natural language) can be quite vague, so it is often quite useful to try and reword things to make matters as explicit and exact as we can.

Suppose next that our local sheriff reports that a motorcycle is in the park. Our question now is this: is the motorcycle in the park illegally? Before reading the next paragraph, pause for a moment, and think about it. What do you think? Is the motorcycle in the park illegally, or not? And *why* did you come up with the answer you came up with? Or, did you conclude that we can't answer the question with the information we have. If so, why not?

Well, let's look at the statute (to be exact, let's look at our reworded Statute 2.34*). According to the statute, if a vehicle is in the park, it's there illegally. But what exactly counts as a **vehicle**? The statute does not say, so we cannot answer this.

To proceed, let's agree on a definition of vehicles. Let us say that, for legal purposes, we will define a vehicle like this:

> **Definition A** (Vehicle):
> A vehicle is anything that has four wheels.

Remark 1.3. Think about **Definition A**. What sorts of things count as vehicles, *according to this definition*? And what sorts of things do *not* count as vehicles? Can you come up with some examples? Try to come up with some outlandish examples too. A good mathematician will always think about limit cases!

Okay, so now there are two statements that we have agreed to: we have agreed to **Statute 2.34***, and **Definition A**. These are our starting points.

Let's return to our question about the motorcycle. Is it in the park illegally? It should be clear now that the answer

is no: the motorcycle is *not* in Hamfordton Park illegally. Why? Because *Statute 2.34** and *Definition A* together say that only four-wheeled things are prohibited, and a motorcycle has two wheels, so it cannot count as a "vehicle" in this situation, *according to the statute and definition that we agreed upon*.

Here is another case. Suppose the sheriff reports that an 18-wheeler is parked in the middle of Hamfordton Park. Is it there illegally? The answer is no, it is not parked there illegally, *according to the definition of vehicle* that we agreed to. An 18-wheeler has, well, 18 wheels, rather than four wheels, so it too does not count as a "vehicle" in this situation.

It may seem counterintuitive to think that an 18-wheeler is not a vehicle. Maybe you think it is also counterintuitive to think that a motorcycle is not a vehicle. But that is not the point.

The point of this exercise is to notice the way that we drew conclusions. We stuck to our agreed upon statute and definition strictly, in order to see what follows logically from them. We concluded *only* what followed strictly from our agreed upon statute and definition.

This kind of reasoning is very much like what happens in math. In math, we always start with some agreed upon starting points, and then we do all our reasoning from there. And we are **strict**, in that we only let ourselves use pure logic to derive conclusions from our starting points. That is the essence of **deductive reasoning**.

Example 1.1. If we strictly follow *Statute 2.34** and *Definition A*, is a wheelchair prohibited in the park? A matchbox car? A bicycle?

Remark 1.4. If you don't think motorcycles or 18-wheelers should be in Hamfordton Park, you might petition the city council to change the legal definition of a "vehicle." Can you think of a definition that would capture the right cases?

1.3 Mathematical Style

IN A MATHEMATICAL INVESTIGATION, coming up with the conclusions (or even coming up with the right starting points) can sometimes be easy, and sometimes difficult. Sometimes

you can see a few conclusions immediately, but other times it can take a lot of creative thinking and time to come up with the most interesting conclusions. There is no mechanical way to do it! It typically takes lots of scratch paper, and lots of patience to sit with the ideas and chew on them, until things become clear.

When a mathematician has finished their work and come up with the conclusions they care about, then they will sit down and write up the results of their investigation. Over the centuries, mathematicians have settled into a particular way of writing, so they can set out their starting points, their reasoning, and their conclusions, in a compact way. There is specific terminology they use to refer to these things too. A mathematician will usually write down their results in the following order:

Terminology. A **definition** stipulates precisely what the meaning of some term or concept is intended to be. An **axiom** is any other statement we should accept as given for the investigation.

Terminology. A **theorem** (or **lemma**) is a conclusion that we have come up with, using nothing but our agreed upon starting points and pure logic.

Terminology. A **proof** is a written description of the logical steps that lead from the given starting points to the theorem (or lemma).

1. **Starting points**. A mathematician will first write down their starting points. Two kinds of starting points are common: definitions and axioms. A **definition** stipulates what the mathematician intends the meaning of some term or idea to be for the course of their investigation. An **axiom** is any statement that the mathematician wants us to accept as given for the course of their investigation

2. **Conclusions**. Next, a mathematician will write down the conclusions that they came up with. Mathematicians call their conclusions **theorems**. Sometimes, they call them **lemmas**. But really a lemma is just a theorem too.

3. **The reasoning**. After each conclusion (theorem or lemma) is stated, a mathematician will write down a short description of the reasoning one should use to derive that conclusion from the given starting points. This written description is called a **proof**. Anybody who wants to follow along in the investigation should be able to read the

proof description, and follow along with the logic, to see how to get to the conclusion too.

So, to sum up, mathematicians will first state their definitions and axioms, then they state their theorems, one after another. After each theorem, they provide a proof, which is a short description of the reasoning that gets you from the starting points to the conclusion (the theorem).

It is important to note that a mathematician will never state a theorem or a lemma without immediately providing a proof for it. If a mathematician suspects that some conclusion follows, but they cannot prove it, then they call it a **conjecture**. So a theorem or lemma is something proved, whereas a conjecture is something that we might suspect to be true, but we have not found a way to prove it, and so we cannot be sure that it is in fact true.

Terminology. Mathematicians never accept a theorem without proof! Without proof, it is just a **conjecture**.

1.4 Internalizing It

READING MATH IS A VERY INTELLECTUAL kind of activity. It is probably not right to say that we *read* math. It is perhaps better to say that we *sit* with it, so to speak, and *chew* on it.

The basic goal when reading math is to **internalize** the ideas. This takes time, and it requires an active thought process (not a *passive* thought process). Definitions, axioms, theorems, and the like are very precise, and very carefully worded. We can't just read over them quickly.

When we read and think about mathematical concepts, what we need to do is stop and read them carefully, break with them down, and work out their meaning and importance slowly. We often have to force our minds to get active in the process, by scribbling things down on scratch paper,

Remark 1.5. It is probably worth noting that mathematicians get very comfortable being in that emotional state of *not understanding something right away*. Mathematical ideas are usually grasped and internalized very slowly.

and trying out examples on our own, to make sure we understand each and every part of the concept.

It often happens that a mathematician will sit with a single definition or theorem for hours, days, or even longer, as they work to understand it. This active process of understanding and internalizing takes time and patience, but the result can be incredibly rewarding.

1.5 Summary

In this chapter, we learned that mathematicians use *deductive reasoning* to do their work.

- **Deductive reasoning** is a process of reasoning where we first agree to some starting points (namely, **definitions** and **axioms**), and then we use nothing but our starting points and **pure logic** to draw further conclusions.
- During our reasoning, we are **strict** in that we do not allow any new information to seep into our considerations. The only thing we can use as we do our thinking is our agreed upon starting points (our accepted definitions and axioms), and pure logic.
- The conclusions that we draw using this deductive process are called **theorems** (or sometimes **lemmas**).
- For any given theorem (or lemma), mathematicians *always* write down a description of the logical reasoning that leads to that theorem (or lemma). This description is called a **proof**.

In this chapter, we did not spend much time talking about how exactly one is supposed to use "pure logic" to deduce

further conclusions from our starting points. What does that actually look like? We also did not talk much about what theorems and proofs look like. In the next chapter, we will begin to look at just those things.

[2]

PROVING SIMPLE FACTS

IN THE LAST CHAPTER, we started to develop a picture of **deductive reasoning**, which lies at the heart of the **mathematical method**. We learned that when a mathematician sets out to perform an investigation of some topic, they establish some basic starting points: some definitions and axioms. Then, they use nothing but pure logic to draw out further conclusions.

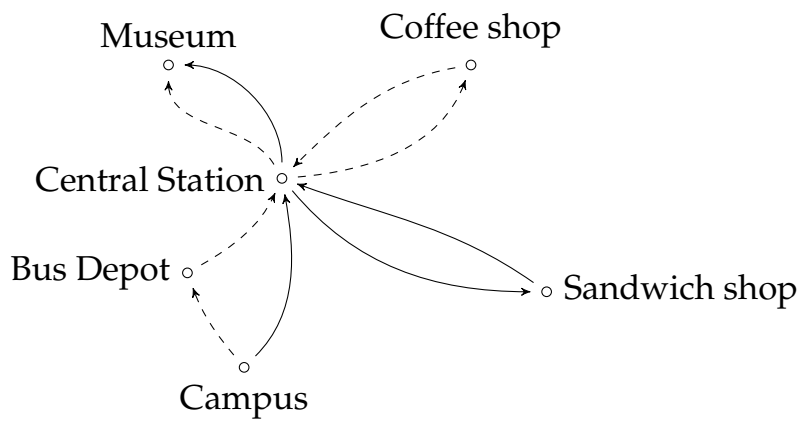
The conclusions that a mathematician establishes through this process are called **theorems** (or sometimes **lemmas**). For every theorem (or lemma) that they discover, a mathematician will always write down a description of the logical reasoning that takes you from the established starting points to the theorem. This description shows you how the theorem actually follows from the starting points, logically speaking. This description of the logical reasoning is called a **proof**.

In the last chapter, we didn't talk much about what this process looks like. And we didn't talk much about what theorems and proofs look like either. When mathematicians use "pure logic" to come up with theorems, and then write down proofs, what exactly does that look like? In this chapter, we are going to start looking at just this sort of thing.

2.1 A Trip to the Museum

LET US BEGIN WITH A MADE-UP SITUATION. Alice, Bob, and Carolina are university students. They want to go to the Museum together, so they agree to meet there later in the day. They will use public transportation. Here is a map of the bus and subway routes:

Bus routes are represented with dashed lines. **Subway** routes are represented with solid lines.



Each leg of a **bus ride** costs $2.50, and each leg of a **subway ride** costs $5.00.

Each ride between two dots on the map costs a fixed amount: $2.50 for the bus, and $5.00 for the subway. So, for example, it costs $5.00 to get from Campus to Central Station by bus (because it costs $2.50 for the bus ride from Campus to the Bus Depot, and another $2.50 for the ride from the Bus Depot to Central Station).

At the start of the journey, Alice has $7.50 pre-loaded on her public transit farecard, Bob has $12.50, and Carolina has $20.00. Each time they take a ride, the cost of that ride is subtracted from their total. So, for example, if Alice takes the bus from Campus to the Bus Depot, $2.50 will be subtracted from her total, and she will be left with $5.00 on her farecard.

In total, **Alice** has $7.50 to spend on fares, **Bob** has $12.50 to spend on fares, and **Carolina** has $20.00 to spend on fares.

It is a simple matter to compute the costs of the various paths these students can take through this map to get to the Museum. In our daily lives, we do this kind of path/money arithmetic all the time. But let's slow down. Let's try to reason as mathematicians about the various travel possibilities.

2.2 The Universe of Discourse

IN MATH, WHEN WE WANT TO REASON about some particular situation, we cut the situation off from the real world, and we imagine putting it into a little universe all by itself. And when we do that, we strip away everything that we don't need or care about.

Terminology. The process of cutting away everything irrelevant and focusing only on the bits we care about is called **abstraction**. By cutting away everything we don't need to think about, we end up thinking only about the relevant bits, **in abstraction** from everything else.

We import our agreed-upon definitions and axioms into our little imaginary universe too. Those definitions and axioms serve as the "laws" that govern our imaginary universe. Much like how the objects around us obey the physical laws of our universe, we can imagine that the objects in our made-up universe obey the definitions and axioms we have stipulated for our investigation.

When we do this, we end up with a collection of things we want to think about, floating in our little universe, obeying the laws that we have imagined. We call the things floating there the **objects** of our investigation, and we call our little imaginary universe the **universe of discourse** (or synonymously, the **domain of discourse**).

In our case, we lift our three students out of the actual city, along with the objects on our transit map (the stations/stops, and bus/subway routes between them). We imagine all of this floating in its own little universe.

The "laws" of our universe are these: the rules of arithmetic (for adding up fares), the stipulation that bus rides cost $2.50 and subway rides cost $5.00, and the amounts of money our students have pre-loaded on their farecards. This is our chosen universe of discourse.

Terminology. The **universe** or **domain of discourse** for an investigation is the collection of **objects** we want to investigate, governed by the **definitions** and **axioms** we have agreed to.

2.3 Proving Theorems

ONCE WE HAVE CONSTRUCTED our little imagined universe, we can then proceed to the task of proving some theorems about our little universe. Essentially, a **theorem** is just a statement that expresses some **fact about our chosen universe**.

To prove a theorem about our chosen universe, we can make use of the objects and "laws" in our chosen universe, and we use pure logic to show that the theorem holds or "is true" in our chosen universe.

There are many ways to prove a theorem. There are some basic patterns though, which we can use to guide us. We'll learn some of the simple patterns here, and then some more complex patterns in the next chapter.

Remark 2.1. Proving a theorem really amounts to showing that the theorem truly holds **in our chosen universe** (it may not hold in other universes). To show this, we can point to various objects in our chosen universe, we can point to how the objects are related, we can point to the "laws" of our chosen universe, and so on. But in the end, we are really just using rigorous logical thinking to show that the theorem must be true, *in* our chosen universe.

2.4 Direct Proof

THE SIMPLEST WAY TO PROVE a single fact about our chosen universe is just to explain directly that it holds in our little universe. We call this **direct proof**.

Let's do an example. One thing you might have noticed about the transit map is that if a student travels from Campus to Central Station, it costs the same, regardless of whether they take the bus or the subway. This is a simple fact about our little universe. So, let us prove it! First, let us state this little fact as a theorem, which we can do like this:

Theorem 2.1. The cost of traveling from Campus to Central Station by bus is the same as the cost of traveling there by subway.

Notice how we introduce the theorem by writing "Theorem" in bold, and then we give it a number. Here I picked 2.1because it is Chapter 2, theorem number 1, but we could pick any unique number or label that we like. We simply add the number so that we can **refer back to it** later. By giving it this label, at any later point in our discussion, we can always say, "see Theorem 2.1."

Notice also that, after providing our theorem with a number, we write out the fact that we want to claim is true, using a **declarative sentence**: we assert that the cost of traveling "from Campus to Central Station by bus is the same as the cost of traveling there by subway." We use a declarative sentence here because in a theorem we are *asserting* or *declaring* that something is true about our little universe.

So this theorem expresses a single fact about our chosen universe: it claims that it costs the same to go from Campus to Central Station, regardless of whether one travels by bus or subway. Of course, now that we have written our theorem down, we need to **prove it**. How do we do that?

I often find it helpful to pretend that I show my theorem to you, and you are skeptical. Then I think, "what could I do to

Remark 2.2. Always *label* your theorems with a unique number, so that you can easily refer back to them later.

Terminology. A **declarative sentence** (synonymns: an **indicative sentence**, an **assertive sentence**, or simply a **statement**) is a sentence which asserts that something is or is not the case. Not all sentences are declarative. The key mark of a declarative sentence is that it can be true or false. If I say "it is raining," that is either true or false, so it is a declarative sentence. But a question — e.g., "is the window open?" — or a command — e.g., "close the window!" — cannot be true or false. You can always tell if a sentence is declarative by asking yourself if it can be true or false. If so, it is a declarative sentence.

prove this to you? What could I do to convince you?" Often, this helps me see at least some of the steps I need to take to convince you that what my theorem says must be true in my chosen universe.

Let's do that here. Let's pretend that I tell you what our theorem says, and you are skeptical. Perhaps you haven't looked at the transit map above. Or perhaps you looked at the map too quickly. Whatever the reason, let's pretend that you are skeptical of my theorem. How Could I prove to you that what my theorem says is in fact true in our little universe?

One thing I could do is calculate the costs of traveling to Central Station by subway and bus, and then I could show you my calculations, and show you that they come to the same amount. That would be a very straightforward way to help you to see that the theorem is true in our little universe.

So let's do that. Let's write up a proof, which describes those very calculations. Here's how we might write it:

Proof. To get from Campus to Central Station by subway, it costs $5.00 for the subway ride. To get to the same place by bus, a student needs to go first to the Bus Depot for $2.50, then to Central Station for another $2.50, which adds up to $5.00. Hence, it costs $5.00 to go by bus too. Either way is the same cost. □

Here we have our first example of a proof. Notice how we indicate the beginning of the proof by writing the word "*Proof*" in italics. Then, we provide a description of our reasoning, which in this case is a description of our calculations. Finally, we signal that the proof is finished with a little box over on the right.

Remark 2.3. By writing out our proof, we communicate exactly why the theorem is true. Anybody who takes the time to check our logic would see that the theorem is correct. There is simply no way that the theorem could be false (in our chosen universe). Of course, one might imagine different subway costs, or different routes. But that would amount to a *different universe*, where our theorem need not apply. Our theorem is a theorem about *our* chosen universe, and our little proof here makes it clear that the theorem is definitely true in our universe; it can't be otherwise.

This is an example of proving a little theorem about our chosen universe. Sure, it is simple, but it constitutes a genuine proof that the theorem holds or "is true" in our imagined little universe. Anybody else could read our proof, check our logic, and see that Theorem 2.1 does indeed hold in our little universe.

2.5 Conjunctions

IN THE LAST SECTION, we showed an example of proving a single fact about our chosen universe. To do it, we used some simple calculations to show directly that the fact must be true.

Sometimes we want to prove not just that a single thing is true in our universe. Sometimes we want to prove that a *combination* of two things are true in our chosen universe.

If we have two statements (call them P and Q respectively), and we want to assert that they are jointly true, we use "and" to combine them. In other words, we take P and Q, and then we put them together to formulate a compound statement that has this shape:

P and Q

Statements with this shape are called **conjunctions**, and we call P and Q the left and right **conjuncts** of the conjunction.

To prove a conjunction, we must prove both of the conjuncts separately. For example, suppose we want to prove that Bob has enough money on his farecard to make it to the Coffee Shop *and* the Museum:

Terminology. A **conjunction** is a statement that has this shape:

P and Q

where "P" and "Q" are replaced by other statements. P and Q are called the **left** and **right conjuncts** of the conjunction.

Theorem 2.2. After going from Campus to Central Station, Bob still has enough money on his farecard to travel to the Coffee Shop, and to the Museum.

Remark 2.4. To **prove** a conjunction, we must prove **both conjuncts** separately. It is not enough to prove only one conjunct. We must show that both are true. Proving that both conjuncts are true separately **is sufficient** to show that the entire conjunction is true.

To prove this, we need to prove both facts: namely, that Bob can reach the Coffee Shop, and also that he can reach the Museum. To prove each such fact individually, we can just use simple calculations. Here is an example of how we might do this:

Proof. To prove the theorem, I need to prove both conjuncts. At the start of his journey, Bob has $12.50 on his farecard. It will cost him $5.00 to get from Campus to Central Station (as proved by Theorem 2.1). So, after traveling to Central Station, Bob will have $7.50 left on his farecard.

- First, I prove the left conjunct. It will cost Bob another $5.00 to get from Central Station to the Coffee Shop and back ($2.50 each way). That will leave $2.50 on his farecard.
- Now I prove the right conjunct. From Central Station, it will cost Bob another $2.50 to get to the Museum by bus, which is exactly how much he has left on his farecard.

Therefore, after traveling to Central Station, Bob still has enough on his farecard to travel to the Coffee Shop and the Museum. □

Notice how in this proof, we announce at the beginning what we need to do to prove the theorem, namely we need to

prove both conjuncts. Then, after providing a little information about how much it costs to get to Central Station, we do indeed prove that both conjuncts are true, with two separate bullet points. In the first bullet point, we show that Bob has enough to get from Central Station to the Coffee Shop and back, and then in the second bullet point, we show that Bob has enough to get from Central Station to the Museum.

Notice also that we used Theorem 2.1 to assert that it will cost Bob $5.00 to go from Campus to Central Station. This is one of the great things about the deductive process. Once we've proved a theorem, it then becomes a known "fact" about our little universe. We can then use it to prove further facts, as we do here.

2.6 Disjunctions

SOMETIMES WE WANT TO PROVE not that two things are true in combination, but rather that either one of two things is true.

If we have two statements (call them P and Q again), and we want to assert that one or the other of them is true, we use "or" to combine them, so as to formulate a compound statement with this shape:

P or Q

Statements with this shape are called **disjunctions**, and we call P and Q the left and right **disjuncts** of the disjunction, respectively.

To prove a disjunction, we only need to prove that one of the disjuncts is true. For example, suppose we want to prove that Bob has enough money on his farecard to travel to the Sandwich Shop, or to the Museum.

Terminology. A **disjunction** is a statement that has this shape:

P or Q

where "P" and "Q" are replaced by other statements. P and Q are each called the **left** and **right disjuncts** of the disjunction.

Remark 2.5. In math, we always assume that disjunctions are **inclusive**, rather than **exclusive**, unless stated otherwise. An exclusive disjunction is true when one of the disjuncts is true, but not both. An inclusive disjunction is true when either *or both* of the disjuncts are true.

Theorem 2.3. After going from Campus to Central Station, Bob has enough money left on his farecard to travel to the Sandwich Shop, or to the Museum.

To prove this, we only need to prove one of the disjuncts. Let's pick the first one: namely, that Bob can reach the Sandwich Shop.

Remark 2.6. To **prove** a disjunction, we must prove **one of the disjuncts**. Proving one of the disjuncts **is sufficient** to show that the entire disjunction is true.

Proof. To prove the theorem, I need to prove one of the disjuncts. I will prove the first disjunct, namely that Bob can reach the Sandwich Shop.

At the start of his journey, Bob has $12.50 on his farecard. It will cost him $5.00 to get from Campus to Central Station (as proved by Theorem 2.1). Then it will cost him another $5.00 to get to the Sandwhich Shop. That adds up to $10.00, which is less than the $12.50 that he started with. □

Remark 2.7. Does Theorem 2.3 say that Bob has enough money to get to the Sandwich Shop or the Museum *but not both*? No, it does add the "but not both" part. Remember: we always assume that disjunctions are inclusive, *unless stated otherwise*. If our theorem did explicitly assert that Bob can go to either one *but not both*, then we would have to show in our proof not only that he could get to one of them, we would also have to show that he could not get to both.

Notice that we did not need to prove that Bob could alternatively go to the Museum. That is because we only need to prove one of the disjuncts to prove the entire disjunction. And indeed: if it is really true that Bob has enough money to get to the Sandwich Shop, then it really is true that he has enough money to get to the Sandwich Shop *or* the Museum.

2.7 Implications

SOMETIMES WE WANT TO PROVE that something is true, provided that some other condition is met. In other words, we

want to say that something is true *on the condition that* something else is true first.

If we have two statements (P and Q again) and we want to assert that Q is true provided that P is true first, we use "if" and "then" to combine them, so as to formulate a compound statement with this shape:

if P then Q

Statements with this shape are called **implications** (or sometimes **conditionals**, because they assert that Q is true on the condition that P is true). We call P the **antecedent** of the implication, and we call Q the **consequent** of the implication.

To prove an implication, we assume that the antecedent P is true, and then we prove the consequent Q. That is to say, we pretend that P is true temporarily, for the duration of the proof, and then we show that once P is in the mix, Q follows.

For example, suppose we want to prove that if a student had $25.00 on their farecard, they would have enough to visit both the Sandwich Shop and the Coffee Shop on their way to the Museum.

Theorem 2.4. If a student had $25.00 on their farecard, they would have enough to visit the Sandwich Shop, the Coffee Shop, and the Museum.

To prove this, we assume that the antecedent is true, so we suppose that a student does have $25.00 on their farecard. Then we show that once they have this amount of money on their farecard, they can reach all the desired places.

Terminology. An **implication** is a statement that has this shape:

if P then Q

where "P" and "Q" are replaced by other statements. P is called the **antecedent**, and Q is called the **consequent**, of the implication.

Remark 2.8. To **prove** an implication, we assume that the antecedent is true temporarily, and then we prove that the consequent follows.

Remark 2.9. Note that no student in our chosen universe actually has $25.00 on their farecard. The antecedent is purely hypothetical. But that's okay with implications! An implication asserts what *would* be the case *if* the antecedent were true.

Proof. Suppose there is a student with $25.00 on her farecard. To get from Campus to Central Station, it costs $5.00 (as proved by Theorem 2.1). It costs another $10.00 to get to the Sandwich Shop and back ($5.00 each way), and another $5.00 to get to the Coffee Shop and back ($2.50 each way). That adds up to $20.00 so far, and that leaves enough for this hypothetical student to get to the Museum ($2.50 by bus, or $5.00 by subway). □

Of course, no such student exists in our chosen universe. But that is fine with implications. What we have proved is not that one of our students *does* have enough to visit the Sandwich Shop, Coffee Shop, and Museum. Rather, we have proved only that *if* a student had $25.00 on their farecard, *then* they would have enough to visit all the stops. That is why we assume that the antecedent is true as the first step in our proof: we are proving only what would follow *if* the antecedent were true.

2.8 Summary

In this chapter, we learned some of the basic techniques that mathematicians use to carry out their investigations.

- We learned that mathematicians like to think of any given situation **in abstraction**, as if the situation were cut off from reality, and stuffed into its own little universe. To construct such a universe, we imagine putting the **objects** we care about into that universe, and we import the relevant **definitions and axioms** that we care about into

it too. In this way, we end up with a nice little universe that has nothing in it but the objects we care to investigate, governed by the "laws" we have agreed to. This little universe is called our **universe** or **domain of discourse**.

- We learned that mathematicians can prove a single fact about their chosen universe through **direct proof**. A direct proof is simply a description of the logical reasoning which shows that the fact in question must **hold** or be **true in** our chosen universe.
- We learned that a **conjunction** is a statement with the shape "P and Q," where P and Q are replaced by other statements. P and Q are called the left and right **conjuncts** of the conjunction. We can prove a conjunction by proving both of the conjuncts separately.
- We learned that a **disjunction** is a statement with the shape "P or Q," where P and Q are replaced by other statements. P and Q are called the left and right **disjuncts** of the disjunction. We can prove a disjunction by proving only one of the disjuncts. We also learned that we always assume a disjunction is **inclusive**, unless stated otherwise.
- We learned that an **implication** is a statement with the shape "if P then Q," where P and Q are replaced by other statements. P and Q are called the **antecedent** and **consequent** of the implication, respectively. We can prove an implication by assuming temporarily that P is true, and then proving that Q follows, once P is in the mix.

[3]

PROVING GENERAL FACTS

IN THE LAST CHAPTER, WE STARTED TO GET A FEEL for the deductive process in detail. We imagined an imaginary universe where three students (Alice, Bob, and Carolina) want to travel to a Museum, using a specified map of bus and subway routes. We learned that this imagined little universe is called the **universe** or **domain of discourse**.

We also learned that we can start proving some **specific facts** about our chosen universe, and we learned some basic techniques for doing that. We learned that we can prove a single, specific fact directly by just describing the logic which shows that it is true in our little imagined universe. This is called **direct proof**.

Then we learned that we can prove a **conjunction** of two facts by proving both of those facts separately, we can prove a **disjunction** of two facts by proving one of those facts, and

Terminology. Something is **more general** if it covers more cases. By contrast, something is **more specific** if it covers fewer cases.

we can prove an **implication** by temporarily assuming that the antecedent is true and then showing that the consequent follows.

In this chapter, we are going to learn a few more techniques that we can use to prove facts about our chosen universe. But this time, we are going to be proving some facts that are **more general** than the more specific sorts of facts we proved in the last chapter. In this chapter, we are going to learn how to prove facts that **range over the entire set of objects** that populate our universe. These are much more general.

3.1 Existential Claims

SOMETIMES WE WANT TO SAY that, among all the objects that populate our chosen universe, at least one of them has some particular characteristic. We usually write such statements so that they have this shape:

there is an x such that x is P

where P is replaced be any sort of assertion about x. We call this an **existentially quantified** statement, or just an **existential claim** for short, because it asserts that there *is* something in our chosen universe (i.e., something "exists" in our chosen universe) which is described by P.

Terminology. An **existential claim** is a statement that has this shape:

> there is an x such that x is P

where "P" is replaced by some assertion about x. Alternatively, one can write this:

> $\exists x$, x is P

which should be read out loud as "there is an x such that x is P." The backwards "$\exists$" reminds us we are asserting that there **exists** an x (in our chosen universe) which is P.

Note that we do not need to explicitly say that x is an object *in our chosen universe*. All we need to say is that "there is an x," and mathematicians will always assume that we are talking only about the xs that live in our chosen domain of discourse.

To prove an existential claim, we need to do two things: (1) we need to bring forward some object from our chosen

universe as an example, and (2) we need to show that our example is in fact accurately described by P. We call such an example a **witness**.

For example, suppose we want to claim that someone in our chosen universe has enough money on their farecard to visit the Sandwich Shop and the Museum:

Theorem 3.1. There is a student x such that x has enough money on their farecard to visit the Sandwich Shop and the Museum.

To prove this, we need to (1) point to one of the students in our chosen universe who is an example of this claim, and (2) show that they do indeed have enough money on their farecard to visit the Sandwich Shop and the Museum.

Proof. To prove the theorem, I provide a witness. Carolina is an example of such a student. At the start of her journey, she has $20.00 on her farecard. To get from Campus to Central Station, it will cost her $5.00 (as proved by Theorem 2.1). It will then cost her another $10.00 to get to the Sandwich Shop and back ($5.00 each way). Finally, it will cost her at a minimum another $2.50 to get from Central Station to the Museum. That adds up to $17.50, which is less than the $20.00 that she started with. □

Remark 3.1. To **prove** an existential claim, we bring forward a **witness** x, and prove that x is in fact accurately described by P.

Notice that we brought forward Carolina as an example, but then we *showed that she does in fact have enough on her farecard to visit the relevant stops*. It is important that we do not forget this second step. We cannot simply say "Carolina is an example," and then stop writing our proof. We need to

complete the second step: namely, we need to show that our witness does in fact satisfy what the theorem says (which in this case means we need to show that she has enough to get the Sandwich Shop and the Museum).

3.2 Universal Claims and Brute Force

SOMETIMES WE WANT TO SAY that *all* of the objects in our chosen universe have some particular characteristic. We usually write such statements with this shape:

for every x, x is P (or synonymously: for all x, x is P)

where P is replaced be any sort of assertion about each x in our chosen universe. We call this a **universally quantified** statement, or just a **universal claim** for short, because it asserts that P applies *universally* to our chosen universe — it asserts that *every* object is P.

In this context, we can speak about *every* x, or *all* x, or *each* x, or *any* x when we say that they are P. All of these expressions mean exactly the same thing. The idea is that we mean to speak about **all** of the objects in our chosen universe — we mean to apply the description P to **each and every one** of them.

To prove a universal claim, we have two options. One option is a method that we call **brute force**. The brute force method is this: go through each object in your chosen universe, one by one, and prove what you need to prove for each one separately.

For example, suppose we want to prove that every student in our universe has less than $22.50 dollars on their farecard at the start of their journey:

Terminology. A **universal claim** is a statement that has this shape:

for every x, x is P

where "P" is replaced by some assertion about each x in our chosen universe. Alternatively, one can write this:

$\forall x$, x is P

which should be read out loud as "for every x, x is P." The upside down "$\forall$" reminds us we are asserting that **all** of the xs (in our chosen universe) are P.

Terminology. A proof of some statement P is done with **brute force** if you manually go through each object in your universe, one by one, and verify that P holds for each one separately.

Lemma 3.1. For every student x, x begins their journey with less than $22.50 on their farecard.

To prove this, we can look at each student in our universe, one by one, and check that they have less than $22.50 on their farecard:

Proof. I will prove the lemma with brute force. Alice starts her journey with $7.50 on her farecard, which is less than $22.50. Bob starts his journey with $12.50 on his farecard, which is less than $22.50. Finally, Carolina starts her journey with $20.00 on her farecard, which is also less than $22.50. Since that covers all of the students, the lemma is proved. □

Suppose we want to prove that no student has less than $7.50 on their farecard, at the start of their journey:

Lemma 3.2. For every student x, x begins their journey with $7.50 or more on their farecard.

We can prove this with brute force too. We can look at each student, one by one, and check that they have $7.50 or more on their farecard:

Proof. I will prove the lemma with brute force. Alice starts her journey with $7.50 on her farecard, which is exactly

Remark 3.2. We are calling our theorem a **lemma** here, rather than a **theorem**, just for fun. A lemma and a theorem are really no different, but sometimes, mathematicians call a theorem a "lemma" to signal that it is not particularly important in its own right, but rather is something they will use later, to help prove other theorems. But really, this is just a matter of taste. "Theorem" and "lemma" are synonyms.

the required amount. Bob starts his journey with $12.50, which is more than $7.50. Finally, Carolina starts her journey with $20.00, which is also more than $7.50. Since that covers all of the students, the lemma is proved. □

If we wanted to, we could be a lot more concise about this proof. Instead of manually listing out each case, we could just write this:

Proof. The lemma can be proved with brute force, simply by checking the starting amounts that each student has on their farecard. □

This is perfectly fine, because any competent reader could flip back a few pages and check for themselves that the initial amounts on the students' farecards, and thereby see that the lemma is in fact true in our chosen universe.

3.3 Universal Instantiation

THE METHOD OF BRUTE FORCE is fine if you have a small enough universe. But if you have a lot of objects in your universe, it is tedious to verify P for every object. And if your universe has an infinite number of objects, you simply *can't* use brute force. You'll never get through all the objects!

So we have another proof method. It is called **universal instantiation**. To prove a universal claim with this method, we do two things: (1) from our chosen universe we pick an **arbitrary object** and we give it an arbitrary name, like "a,"

Terminology. A proof of a universal statement is done by **universal instantiation** if you prove it for an arbitrarily chosen object in your domain.

and then (2) we show that P is actually true of this object a, **without appealing to any individual features** of a.

Let's spend a little time unpacking this idea. What does it mean to pick an *arbitrary* object a from the domain? And what does it mean to say that we *do not appeal to any individual features* of a?

Basically, it means we put on a blindfold for the duration of the proof, and then we let someone else pick an object at random from our universe, without telling us which one they've picked. We don't know its real name, so we just call it "a." This way, we don't know *which* object got picked, and because we don't know which object got picked, we also don't know any *individual details* about it (not even its name).

Now, if you like, you can actually wear a blindfold, and then have someone randomly pick an object from your domain, which you will refer to as "a." But obviously we don't need to do that. All we really need to do, to pick an arbitrary object a, is just write down something like the following at the beginning of our proof:

> Let a be an arbitrary object from our chosen universe.

When we write something like that, mathematicians understand that what we mean is this:

> You (the reader) should pick some object from our chosen universe, it doesn't matter which one, and let's just call it "a" for the duration of our proof. I won't look, so that I won't know which one you've picked.

So by writing down "let a be an arbitrary object," we are in effect allowing an object from our universe to be picked at random, and we are stipulating that we will refer to it as "a." We are "blindfolded" because we don't know which object

Remark 3.3. It is important that the name "a" be chosen arbitrarily, so that it does not signal which object got picked. In essence, it should be a temporary fake name that hides the identity of its bearer. As an analogy, think about news sources that are reported under fake names, for protection. It would do no good to report such sources under a name that told us who they really were! The name "a" should be arbitrary like this. So, for example, don't pick "a" if there is an object named "a" in your universe! Use some other name, which doesn't belong to an object already. Say instead something like, "let b be an arbitrary object from our chosen universe," or "let c be an arbitrary object from our chosen universe."

got picked. All we did was give it the temporary name "a," so we have some way to refer to it, for the duration of our proof. But beyond that, we know nothing about which object this "a" is supposed to be.

As an analogy, imagine that you are a student in a classroom. You are blindfolded, standing on one side of the room, and your classmates are on the other side of the room. You say to your classmates, "pick one of you to step forward, it doesn't matter who, but don't tell me who. Whoever you are that gets picked, I'll just call you 'a' for now." Suppose that the students then draw straws, and the chosen one steps forward.

When we write "let a be an arbitrary object in our universe," we are doing something just like this. We are blindly selecting an object from our universe, without knowing which one is selected. And, since we don't know which one it is, we will use a temporary name "a" to refer to it for the duration of our proof.

Notice also the following. Since we don't know which of the objects in our universe got picked, we don't know any **specific** or **individual details** about it. Think to the classroom again. If we don't know who was picked, we don't know if they have brown hair, if they are tall, and so on. We simply don't know any of their identifying features.

Remark 3.4. If we don't know which object a is, we don't know any individual details about a. What do we know then? We know (1) anything that is true of all our objects, (2) we know any information contained in our starting points, and (3) we know any information contained in any theorems (or lemmas) we have proved already.

Now, since we don't know which particular object got picked, and since we don't know any specific details about the object that was picked, we are left with a question: what exactly *do* we know about our pick? Well, when we're blindfolded, we only know things that are true of all of the objects in our domain.

Imagine the classroom again. If you don't know who a is, the only facts you know about them are facts that are true of all the students in the class. For example, you may not know if they have brown hair, but you do know that they are

a fellow student in your class, you know that they attend the class at the same time as the other students each day, perhaps every student has an assigned seat, and so on.

We also have our **starting points**, namely the definitions and axioms we started with. And if there are any **theorems** (or lemmas) we have proven already, then we have them too. All of this information is available to us. The only things we *don't* know are individual details about our chosen object, because we simply don't know which object it is.

With all that in hand, what we do next is this: we take all of this general information that we know about our objects and our chosen universe, and we use it to prove that a is truly described by P. We show that the object a, which was picked at random, *must* in fact be P.

So to summarize, the pattern for universal instantiation is this. The claim we want to prove is that every x in our domain is P. To prove this, we pick an arbitrary a from our domain, and then we take general information that we know about our universe and all of our objects, and we use that to prove that a is in fact truly described by P.

Let's do an example. Suppose we want to claim that every student in our chosen universe starts the journey with enough money on their farecard to reach the Museum:

Remark 3.5. To **prove** a universal claim using **universal instantiation**, pick an *arbitrary* object a from your chosen universe, and then show that a is P using only the general information that is available.

Theorem 3.2. For every student x, x begins the journey with enough money on their farecard to reach the Museum.

Let us prove this using the method of universal instantiation. To do that, we first pick an arbitrary student from our domain. We do this blindfolded, so we don't really know which one gets picked. Hence, we'll just call the selected student "a."

Next, we need to show that the chosen student does have enough money on their farecard to reach the Museum. However, we cannot appeal to any specific details about our chosen student.

On the contrary, we can only appeal to things that are true about all of our students. What sorts of things do we know about all of our students? Well, at this point, we have our starting points. But we also have two lemmas, and in particular we know Lemma 3.2: that every student starts the journey with $7.50 or more on their farecard.

That's enough information to prove that the student we are calling "a" — whoever they may be — has enough money on their farecard to reach the Museum:

Proof. To prove the theorem, let a be an arbitrary student. From Lemma 3.2, we know that every student starts their journey with at least $7.50 on their farecard. That amount is just enough to get from Campus to the Museum: for it costs $5.00 to get from Campus to Central Station (as proved by Theorem 2.1), and it costs another $2.50 to go by bus from Central Station to the Museum. Therefore, a begins the journey with enough on their farecard to reach the Museum, whoever a might be. Since a was chosen arbitrarily, it follows that every student begins the journey with enough money on their farecard to reach the Museum. □

Remark 3.6. Notice how we prove that *one* (arbitrarily chosen) student can reach the Museum, and then we conclude that our theorem therefore holds for *every* student. But how are we justified to conclude that what holds for just *one* student holds for *all* of them? The answer lies in the fact that we stay blindfolded the whole time. By staying blindfolded, what we are doing is coming up with a way to show that any student, whoever they might be, has enough money to reach the Museum. Whichever student we pick, it doesn't matter, because we can just repeat the same proof for *them*, and it would still work, because we aren't appealing to anything specific about our pick. So in effect, we come up with a proof that we can repeat for each and every object in our universe, and that is why it proves the entire universal claim.

Notice in this proof that we followed the pattern for universal instantiation. We first picked an arbitrary student, who we called a. We didn't specify anything about who it might be, so it could be Alice, it could be Bob, or it could be Carolina. We don't know which one it is, nor do we care. Then, we used information that we know about all of the students,

to show that whoever a might be, they will most certainly have enough on their farecard to reach the Museum.

Even though we proved this fact about *one* student, we were careful in picking this particular student arbitrarily (hence, it could be any one of them). So what we proved here really turns out to be a proof that *any* student has enough money on their farecard to reach the Museum, which is precisely what our theorem asserts.

3.4 Summary

In this chapter, we learned some techniques for proving more general facts about our chosen universe. In particular, we learned how to prove facts that range over all of the objects in our universe.

- We learned that an **existential claim** is a statement with the shape "there is an x such that x is P." This asserts that among all the objects in our chosen universe, there is at least one who is P. To prove an existential claim, we provide a **witness**, which means that we bring forward an object from our chosen universe as an example, and we show that they are in fact accurately described by P.
- We learned that a **universal claim** is a statement with the shape "for every x, x is P." This asserts that every object in our chosen universe is P. To prove a universal claim, we can either use **brute force**, or we can use **universal instantiation**, i.e., we pick an arbitrary object a from our chosen universe, and then we show that a is in fact truly described by P, using only facts that are true about all of our objects.

[4]

PROVING NEGATIONS

IN THE LAST CHAPTER, we learned how to prove facts that range over the entire collection of objects in our chosen universe. We learned that an **existential claim** asserts that some fact is true of at least one of the objects in our chosen universe, and we learned that to prove an existential claim, we need to provide a **witness**.

We then learned that a **universal claim** asserts that some fact is true of every object in our chosen universe. We learned that we can prove such claims by **brute force**, which means that we prove the fact about each object in our universe separately.

Alternatively, we can prove a universal claim by **universal instantiation**, which means we pick an arbitrary object from our universe and show that the fact holds for that arbitrarily chosen object. Since we picked the object arbitrarily, this

means that we have in effect proven that the fact holds for *any* (and hence *every* object in our universe).

So far, we have been dealing with what we might call *positive statements*, i.e., statements which assert that something *is* the case in our chosen universe. However, we can also formulate statements which assert that something is *not* the case in our chosen universe. These kinds of statements are called **negations**. We might say that negations are statements which assert *negative facts*, i.e., they assert that some particular fact does *not* hold true in our chosen universe.

Of course, if we want to assert that a negation is true in our chosen universe, then we need to prove it, just as we do for any other assertion we want to make about our chosen universe. But how exactly do you prove that something is *not* the case? In this chapter, we will talk about negations, and learn some of the ways to prove that a negation is true in one's chosen universe.

Ponder. How might you attempt to prove a negative fact? For example, how might you prove that there is no Santa Claus? How might you prove that when you drop a heavy object, it never floats upward?

4.1 Simple Negations

AS WE NOTED A MOMENT AGO, sometimes we want to prove not that something *is* the case in our chosen universe, but rather that something *is not* the case.

If we have a statement (call it P), and we want to assert that P is not true, we can combine P with "it is not the case that" (or just "not" for short) to formulate a compound statement with this shape:

not P (synonymously: it is not the case that P)

Statements with this shape are called **negations**. The words "not" or "it is not the case that" are the words that signal or do the negating, and P expresses the fact being negated.

Terminology. A **negation** is a statement that has this shape:

not P

where "P" is replaced by another statement.

In English, "not" usually doesn't appear at the front of the sentence. Instead, it is much more common for "not" or "does not" to appear somewhere in the middle of the sentence (usually before a verbal phrase). For example, we might say:

> Alice does not have enough money on her farecard to visit the Sandwich Shop.

Notice that "does not" appears before the verbal phrase "have enough money." It is not at the front of the sentence. However, we can usually reword such sentences so that "It is not the case that . . . " does appear at the front of the sentence. Hence, we can reword the aforementioned sentence like this:

> It is not the case that Alice has enough money on her farecard to visit the Sandwich Shop.

For our purposes here, it usually does not matter much which way we word the sentence. For most of the cases that we will discuss, either wording has the same meaning.

How do you prove a negation? If the negation is about a particular object in your chosen universe, you can just verify that the negation applies to that object **directly**. For example, suppose we want to prove that Bob does not have enough money on his farecard to visit the Sandwich Shop and the Museum.

Theorem 4.1. At the start of his journey, Bob does not have enough money on his farecard to visit the Sandwich Shop and the Museum.

To prove this therem, we can just do the calculation directly, to show that Bob doesn't have enough to reach both the Sandwich Shop and the Museum:

Remark 4.1. If you find yourself confused about the meaning of a negation, try to reword it so that "It is not the case that . . . " appears at the front of the sentence. Sometimes, this helps us see more clearly exactly what it is that is being negated.

Remark 4.2. We could reword our theorem so that it has this form:

> It is not the case that, at the start of his journey, Bob has enough money on his farecard to visit the Sandwich Shop and the Museum.

This has the same meaning as what we stated in our theorem.

Proof. At the start of his journey, Bob has $12.50 on his farecard. It will cost him $5.00 to get from Campus to Central Station (as proved by Theorem 2.1). If he goes to the Sandwich Shop and back, it will cost him another $10.00 ($5.00 each way). Since Bob does not have $15.00 on his farecard, he can't make it to the Sandwich Shop and back. And since he doesn't have enough money to get to the Sandwich Shop and back, he certainly doesn't have enough to then make it to the Museum. □

Notice that we proved a negative fact here. We didn't prove that Bob *has* enough money to get somewhere. We proved that Bob does *not have* enough money to get somewhere. So, we proved a negation.

But also, notice how we proved it. We showed that this negative fact must be true by showing it directly, with simple calculations. This is the same technique we used to prove Theorem 2.1. Simple negations can thus be proved like any other simple fact about our chosen universe. In particular, we can just use **direct proof** to show that the theorem must be true in our chosen universe.

4.2 Brute Force

NOW SUPPOSE WE WANT TO PROVE A NEGATION that is a little more complicated than the simple negation we looked at a moment ago. Suppose, for example, that we want to prove that *no student* begins the journey with enough on their farecard to visit the Sandwich Shop, the Coffee Shop, and the Museum. How do we prove this?

One option is that we could do it by *brute force*. That is, we could go through all of the students in our chosen universe, one by one, and show that in each case, the student in question does not have the funds to visit all the stops.

As with brute force and universal claims, this is fine if our universe is small enough. But if our universe is large enough, it becomes tedious to verify the negative fact about each object in our universe. And of course, if our universe has an infinite number of objects in it, then we simply can't use brute force, because we'll never get through all the objects.

4.3 Proof by Contradiction

FORTUNATELY, THERE IS ANOTHER METHOD we can use to prove negations, which is called *proof by contradiction*. This work for many cases.

To carry out a **proof by contradiction**, we do two things. (1) First, we begin by assuming hypothetically that the *opposite* of what we want to prove is true. That is, we start the proof by pretending that the opposite of what we are trying to prove is true. (2) Then, we use pure logic to prove that this leads to a contradiction. In other words, we show that if we pretend that the opposite of what we want to prove is true, we end up in a contradictory state of affairs. If we succeed in doing that — if we succeed in showing that the opposite of what we want to prove does indeed lead to a contradiction — then we can conclude that the original thing we wanted to prove must be true.

Terminology. To perform a **proof by contradiction**, assume hypothetically that the opposite of what you want to prove, and then show that this leads to a contradiction. Once you have derived a contradiction, you can conclude that the hypothetical assumption you started with must be false, and therefore the original thing that you wanted to prove must be true.

Why does this work? To see why it works, let's be explicit about each step of the process:

1. We start with a hypothetical assumption: namely, the opposite of what we want to prove.

2. Then, we take a series of steps (using pure logic) to show that this leads to a contradiction.
3. When we reach a contradiction, that is a signal that we have taken a wrong step somewhere, because a contradiction cannot ever be true.
4. So, we need to backtrack, and retrace our steps. We must examine each step of our proof, to see where the error is.
5. Since we used nothing but pure logic for each of the intermediate steps in our proof (and assuming that we in fact made no mistakes during each such step in our proof), then we will find that none of those steps are wrong.
6. Therefore, the only thing left that could be wrong is the hypothetical assumption we made at the beginning, namely the opposite of what we originally wanted to prove.
7. Since *that* can't be true, the *opposite* of it must be true, which is the original thing that we wanted to prove.

Let's do an example. Let us prove that no student has enough money on their farecard to visit the Sandwich Shop, the Coffee Shop, and the Museum.

Theorem 4.2. At the start of the journey, no student has enough money on their farecard to visit the Sandwich Shop, the Coffee Shop, and the Museum.

To prove this, we first assume hypothetically that the opposite is true. What is the opposite? Well, the theorem states that there is *no* student with enough money on their farecard to visit all the stops. So the opposite of that is that *there is* a student with enough money to visit all the stops.

Remark 4.3. One presupposition that underlies proof by contradiction is that a contradiction cannot ever be true. I cannot ever exist and not exist at the same time, you cannot ever be a human and not a human at the same time, etc.

Another presupposition that underlies proof by contradiction is the idea that if one of a pair of a contradictories is false, then the other must be true. If it is false that I am *not* a human, then it must be the case that I *am* a human.

Hence, we will start our proof by pretending that there is such a student, and then we will carefully take steps to show that this leads to a contradiction.

Proof. I prove the theorem by contradiction.

1. Suppose there is a student a who, at the start of the journey, has enough money on their farecard to travel to the Sandwich Shop, the Coffee Shop, and the Museum.
2. Student a will spend \$5.00 to get from Campus to Central Station (as proved by Theorem 2.1).
3. Student a will spend \$10.00 to get from Central Station to the Sandwich Shop and back (\$5.00 each way).
4. Student a will spend \$5.00 to get from Central Station to the Coffee Shop and back (\$2.50 each way).
5. Student a will spend a minimum of \$2.50 to get from Central Station to the Museum.
6. Therefore, at the beginning of the journey, student a would have to have \$22.50 or more on their farecard.
7. But at the beginning of the journey, every student has less than \$22.50 on their farecard (as proved by Lemma 3.1).

Since (6) and (7) contradict each other, the assumption (1) must be false, and its opposite must be true. □

Let's go over each step of this carefully. We begin in step (1) by assuming hypothetically the opposite of what we want to prove, namely that there is a student (who we call a) who

has enough money to visit all the required stops. Then, in steps (2) through (6), we enumerate all the fares required to visit the relevant stops, concluding that student a would have to begin the journey with $22.50 or more on their farecard. However, we point out in step (7) that every student has less than $22.50 (which we already proved in Lemma 3.1), and that contradicts (6). This is a contradiction, which simply can't be true. It's an impossible state of affairs. You can't both have less, and not have less, than $22.50 on your farecard at the beginning of the journey.

So something has gone wrong. We must have made an error, because we ended up in a contradiction, i.e., an impossible state of affairs. We need to go backwards, and check each step in our proof, to find our error. Let's do that:

- Let's check step (7) first. Is it correct that every student has less than $22.50 on their farecard? Yes, that is correct. We proved it in Lemma 3.1, and a cursory glance at the original totals on the farecards reveals that indeed, the most that any student has on their farecard when they begin their journey is $20.00. So it is correct that every student has less than $22.50 on their farecard.

- Since step (7) isn't the problem, let's go backwards, and check step (6). Is it correct that our hypothesized student a would have to begin the journey with $22.50 or more on their farecard, if they are to visit all of the relevant stops? Yes, that is correct too. This is the correct sum (we can even check it on a calculator).

- Since step (6) isn't the problem, let's go back one more step, and check step (5). Is it correct that $2.50 is the minimum amount student a would have to spend to get from Central Station to the Museum? Again, this is correct. We can look at the original map to see that there

Remark 4.4. This careful process of going backwards through our proof and examining each step is here to show us that each and every step in our proof, except for assumption (1), is firmly grounded in the facts about our chosen universe. The validity of each step is either grounded in some true fact about our chosen universe (like a subway or bus route on the map), or it is grounded in one of the agreed upon laws that govern our universe (like the cost of bus rides, or the arithmetic we use to add up the total costs). So the *only* place we could have gone wrong in this proof is assumption (1). That's the only place where an error could be.

are only two ways to get to the Museum from Central Station: by bus (which costs $2.50), or by subway (which costs $5.00), and the $2.50 option is the cheapest. So step (5) is correct too.

- Okay, then let's check step (4). Is it correct that student a would have to spend $5.00 to get from Central Station to the Coffee Shop and back? Yes, it is correct. We can see it by checking the map.

- Let's check step (3). Is that correct too? Again, the answer is yes. We can see on the map that student a would have to spend $5.00 to get from Central Station to the Sandwich Shop, and $5.00 to get back to Central Station.

- Let's check step (2). Is it correct that student a would have to spend $5.00 to get from Campus to Central Station? Again, the answer is yes. We proved this already in Theorem 2.1, and we can also just look on the map to see that this is correct.

- So that leaves step (1), which we assumed to be true hypothetically to start with. Is this one correct? Here, we must conclude that it cannot be correct, because we just confirmed that all the other steps are logically correct. The only step in our proof that could possibly be wrong is therefore this first hypothetical assumption. So this assumption — that there *is* a student with enough fare money to visit all the relevant stops — must be false.

- So, if (1) is false, then the opposite of what it says must be true. Since (1) asserts that there *is* a student with the relevant funds, and that was found to be something that is impossible in our chosen universe, then it follows that the opposite must be true, namely that *no* student has the relevant funds.

Remark 4.5. In effect, a proof by contradiction shows that our assumption (1) is *incompatible* with the other facts of our chosen universe. It shows that it is *impossible* for assumption (1) to be true, *along with* the other facts we know about our universe. That is what the contradiction reveals.

Proof by contradiction always follows this same pattern. We begin by pretending that the opposite of what we want to prove is true, and then we show that this leads to a contradiction. By doing this, we show that the what we pretended at the start is an impossibility in our chosen universe. It is something that is simply incompatible with the facts of our chosen universe. Since it is impossible, and cannot be true, it follows that the opposite must be true, which is what we originally wanted to prove.

4.4 Summary

In this chapter, we learned a few techniques for proving negative facts about our chosen universe.

- We learned that a **negation** is a statement with the form "not P" (or more explicitly, "it is not the case that P").
- We learned that if we want to prove a simple negation, we can just prove it directly, using **direct proof**.
- We learned that if we want to prove a more general negation, we can use **brute force**, and show that the negation applies to each object in our universe, one by one.
- Or, we can use **proof by contradiction**. To perform a proof by contradiction, we begin by pretending that the opposite of what we want to prove is true, and then we show that this leads to a contradiction. Since no contradiction can be true, it follows that what we initially pretended was true is actually impossible in our chosen universe, and therefore the thing we wanted to prove originally must be true.

[5]

FURTHER READING

To pursue matters of the mathematical method further, the following list may offer some helpful starting points.

When reading and studying any topic in math, it can sometimes be best to work through multiple texts in parallel. Work on one until you get stuck, then go to the other. Then back to the first, and so on. And always try to do the exercises.

- Wilder (2012, chs. 1–2) provides an overview of the mathematical method (as well as many other interesting topics relevant to the mathematical approach).
- Stewart and Tall (2015) offers a good introduction to mathematical logic too.
- Devlin (2012) provides a simple overview of basic mathematical logic.

- Blackburn and Bos (2005(@) gives a simple introduction to logic from the perspective of computational linguistics. If the mathematical way of presenting logic feels impenetrable, try this one and the next.
- Dowty et al. (1981, chs. 2–3) offers another but more thorough introduction to logic from the perspective of formal linguistics. This might also be helpful if the mathematical way of presenting logic feels too curt.
- Teller (1989) is one of the best introductions to modern formal logic that I know of. If there is one text to learn logic from, I would recommend this one. If you can find it in print for an affordable price, great. There is nothing better than having the text at hand, to underline, highlight, scratch notes in the margins, and so on. It is also freely available from the author, at https://tellerprimer.ucdavis.edu/

For further introductions to modern mathematical topics as a whole, see the following, which cover many of the topics we cover in this book, and more:

- Steinhart (2018) provides an accessible introduction to a variety of mathematical topics.
- Burger and Starbird (2010) is an extremely accessible introduction to many mathematical topics.
- Stewart (1995) is an old classic, but it does discuss a variety of mathematical topics in an accessible manner.

For detailed guidance on writing mathematical proofs:

- Velleman (2019) provides an excellent tutorial on the business of proper proof writing. Doing all the exercises in this book is well worth it.
- Wolf (1998) is another fairly accessible guide to the business of proper proof writing.

- Madden and Aubrey (2017, chs. 7–8) provides a detailed discussion of logic and proof writing that can be tackled after working through Velleman (2019).

Part [2]

SETS

One of the most basic ways to add structure to an otherwise unorganized heap of objects is to put the objects into collections (i.e., put them into various buckets). In math, we call a collection a **set**.

In Part 2, we discuss **sets** and many of the various things we can do with them.

[6]

COLLECTIONS

ONE OF THE WAYS WE ORGANIZE our thinking is by gathering things into collections. We are so good at it that we often don't notice we are doing it. But we do. And grouping things together has a purpose: it lets us think about the objects all together as one thing — as one blob, so to speak.

Moreover, we can do a variety of things with collections. We can compare them. We can combine and divide them, to form new collections. We can even make collections of collections, and collections of collections of collections.

Recall that math is concerned with the abstract study of structures. Well, sorting things into buckets imposes some structure on an otherwise unstructured bunch of objects. So let us say that **collections** are a minimal kind of **structure**, at least insofar as they function something like a container that we can put things in.

Ponder. What kinds of collections do you form in your mind, as you go about your day? What kinds of collections help you think through problems better?

Mathematicians have formalized the primary ways that we work with collections, and they have boiled it down to the fundamentals. The resulting theory is simple, and basic. Mathematicians call a collection a **set**, and the theory built up around sets is called **set theory**.

Terminology. **Set theory** is the mathematical theory of *sets*. A **set** is just a collection of objects.

In this chapter, we will introduce the basic ideas and definitions behind set theory. In the following chapters, we will turn to some of the things we can do with sets.

6.1 Forming Sets

TO FORM A SET, take any number of objects, and announce, "let's consider all of these things together, as one set." For example, I can point to the pencil, book, and coffee mug on my desk, and say, "those three things — let's consider them together, as one set."

Of course, I could physically put those objects into a container (a bag, or a box), but that is unnecessary. I can just collect them together *in my mind*. That is all I really need to do to form a set.

I can put almost anything into a set, and the objects don't need to be related. I can consider all the ideas I had yesterday as one set, or a jar of peanut butter, the number four, and all 1980s teen coming-of-age movies.

Terminology. A set is **well-defined** if it is clear and unambiguous which objects belong in the set and which do not.

The only basic restriction on forming a set is that it must be **well-defined**. When we say a set is "well-defined," we mean it is clear which items belong in it, and which do not.

Suppose I say, "consider the set of the biggest stars in the galaxy." That would not be well-defined, because it is not clear exactly how big a star needs to be to belong in the set. I can fix the problem by stipulating some particular mass, and saying that a star belongs if it has at least that mass.

On rare occasions, we try to form sets which are in fact **impossible** to form. Such sets are ill-defined (obviously). But usually this only occurs if we try to get too clever.

As an example, consider the following. An adjective is *autological* if it possesses the property it describes, otherwise it is *heterological*. For example, "English" is autological, because it is itself an English word.

Now suppose I say, "consider the set of all heterological adjectives." Many adjectives would clearly belong in this set. But what about "heterological" itself? We can't answer this without ending up in a contradiction.

We have to be careful about this sort of thing, but we usually only need to worry about it in paradoxical cases, like the hetorological example just mentioned.

Terminology. A set is **impossible to form** if trying to form it results in a contradiction. Contradictions signal an impossible state of affairs.

Remark 6.1. Is "heterological" heterological? If we say "yes, 'heterological' is heterological," then it would describe itself after all, which contradicts its meaning. If we say "no, 'heterological' is not heterological," then it follows that it would be autological, which again contradicts its meaning.

6.2 Set-Rosters

IF WE WANT TO WORK WITH SETS, we need to tell others about the set we have in mind. In particular, we need to be able to specify *which* items go in the set. When we announce what goes in a set, we say that we **specify** the set.

One way to specify a set is to just list out all its objects. For example, I can specify the set of objects on my desk:

the pencil, the book, the mug

But when we list the objects in a set, we should always wrap the list in curly braces. Like this:

{ the pencil, the book, the mug }

When we list out the objects in a set like this, with curly braces around the list, we call this **set-roster notation**. We

Terminology. To **specify** a set is to specify which objects the set is made up of. That is, we specify which objects go in the set.

Notation 6.1. **Curly braces** are the universal sign to mathematicians that we are dealing with a **set**.

Terminology. To specify a set using **set-roster notation**, simply list the objects between curly braces.

call it "set-roster" because by listing out the items, we are providing a *roster* of the items in our set.

What do we do if we want to list out a really big set? For example, suppose I want to write out the set of numbers from 1 to 100. How would I do that in set-roster notation?

I could write them all out, one by one, but that'd be long and tedious. Instead, we can use an ellipsis (three dots) to indicate that there is a pattern. So I could do it like this:

$$\{1, 2, 3, \ldots, 100\}$$

Notation 6.2. In set-roster notation, an **ellipsis** denotes a repeating pattern.

You can read this aloud like this: "this is the set containing the numbers 1, 2, 3, and so on, up through 100."

We can use this technique to specify infinite sets too. Suppose I want to specify the set of positive even numbers: 0, 2, 4, and so on infinitely. I could write that out like this:

$$\{0, 2, 4, 6, \ldots\}$$

You can read this aloud like so: "this is the set containing the numbers 0, 2, 4, 6, and so on (forever, into infinity)."

6.3 Naming Sets

WHEN WE TALK ABOUT a particular set, we can end up referring to it again and again. It's tedious to write out the set-roster every time we need to mention it. Instead, we can give our set a name.

Remark 6.2. When we use an equals sign to indicate a name, it has the following meaning: it says that the thing on the left side of the equals sign is the same as (or is a name for) the thing on the right side of the equals sign.

For example, I might call the objects on my desk my "Desk Kit." In mathematical notation, I would write this:

$$\text{Desk Kit} = \{ \text{ the pencil, the book, the mug } \}$$

You can read this aloud like so: "let 'Desk Kit' be the set containing the pencil, the book, the mug."

Mathematicians prefer to name sets concisely: with just one italicized uppercase letter. In keeping with this practice, let's name my set A:

$$A = \{ \text{ the pencil, the book, the mug } \}$$

You can read this aloud like so: "let A be the set containing the pencil, the book, and the mug."

Mathematicians also prefer italicized lowercase letters as names for objects. Let's give our objects such names:

Remark 6.3. We can specify short names for any of the objects in our universe of discourse, using this same format.

$$\begin{aligned} p &= \text{the pencil} \\ b &= \text{the book} \\ m &= \text{the mug} \end{aligned}$$

You can read this aloud like so: "let p be the pencil, let b be the book, and let m be the mug." Then we can specify our set like this:

$$A = \{p, b, m\}$$

You can read this aloud like so: "let A be the set containing p, b, and m." Notice how much more concise this is, without losing any essential information.

6.4 Set Membership

HERE IS SOME VOCABULARY. We call the objects in a set the **elements** of the set. If an object is an element of a set, we say

Terminology. The objects in a set are called its **elements**. If an object is an element of a set, we say it is a **member** of the set.

it is a **member** of the set. If we want to say that an object is a member of a set, we use a special symbol: the $\in$ symbol.

For example, suppose we want to say that p (the pencil) is a member of my desk kit set A. We would write it like this:

$$p \in A$$

You can read this aloud like so: "p is a member of A," or even "p is in A." If it helps, you can think of the $\in$ symbol as being a little "e" for *element*, and so you can read it like this: "p is an *element* of A."

If we want to say that some object is *not* in a set, we use the $\in$ symbol again, but we put a slash through it, like this: $\notin$. For example, if we want to say that a ruler (let's call it "r") is not in my desk kit set A, we can write this:

$$r \notin A$$

You can read that out loud like this: "r is not an element of the set A," or even more concisely: "r is not in A."

Notice that when we use the $\in$ or $\notin$ symbols to say that something is or is not a member of a set, we make a **statement**. That is, we issue a declarative sentence (using mathematical symbols), and that sentence can be true or false.

And of course, there is nothing to prevent us from asserting something false. For example, I can assert that r (the ruler) is in A:

$$r \in A$$

But is this a true statement? No, it is not. We know from our roster that A contains p, b, and m, but no r. So this statement is false. Similarly, consider this statement:

$$p \notin A$$

Notation 6.3. To denote that an object b is in a set C, we write this:

$$b \in C$$

That should be read as "b is in C."

Notation 6.4. To denote that an object b is not in a set C, write this:

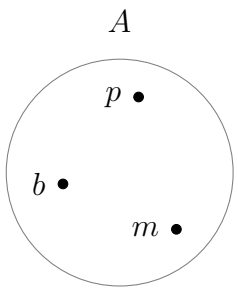
$$b \notin C$$

Read that as "b is not in C."

Remark 6.4. Writing (or saying) that some element is (or is not) a member of a set is a statement. It asserts something, which can be true or false. And there is nothing preventing us from writing (or saying) that an object is in a set when in fact it is not, nor is anything preventing us from writing (or saying) that an object is not in a set when in fact it is.

This asserts that the object p (the pencil) is *not* in the set A, but is that true? No, it is not. We know from our roster that p is in fact a member of A.

Another way to think about a set is this: a set is just a bag or container of items. Hence, our set A which contains the elements p, b, and m can be pictured as a bag of the items:



Here we have pictured the elements of the set as **points**, labeled with their names. The set itself is pictured as a circle that is drawn around the points. So this visually shows us that a set is **container** that contains some points (elements). Throughout, we will refer to the members of a set as its **elements** or its **points**, but we will mean these as just two different ways of talking about the same thing: we really just mean the objects that are contained inside the set.

Remark 6.5. A set can be pictured as a **bag** or **container** with some **points** scattered about inside it.

Terminology. We will sometimes call the members of a set its **elements**, but other times we will call them **points**. We use these as synonyms, as two different ways of talking about the same thing.

6.5 Set-Builder Notation

WE HAVE BEEN SPECIFYING SETS with set-roster notation. But there is another way we can specify a set: we can provide a description or a recipe that tells the reader how to build the set themselves.

For example, I might say to you, "look around the environment you are in, take every object which is green, and put

Remark 6.6. There are two ways to specify a set:

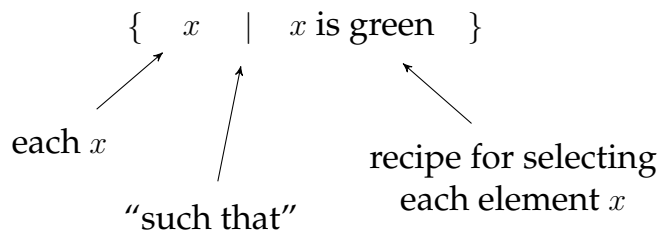
1. List out the elements (the roster of items)
2. Provide a recipe for building the set

it into a set." By doing that, I tell you how to build the set, without listing out the roster.

If I want to write this out, I would write it like this:

$$\{\ x \mid x \text{ is green}\ \}$$

You can read this aloud like so: "this is the set that contains every x which is such that x is green." This might seem a little obtuse, so let's break it down:



Note the following about this notation:

- We surround the whole thing in curly braces, because that is the universal sign that we are dealing with a set.
- In the middle, we write down a vertical bar, which stands for "such that."
- On the left side of the bar, we write down a lowercase letter (usually x) to stand for each object in the set.
- On the right side of the bar, we specify the recipe which tells us how to select the elements that go in the set.

This notation might seem cumbersome, but it allows us to specify pretty much any set we like, no matter how complicated or large it might be. We call this **set-builder notation**. It has the shape:

$$\{\ x \mid x \text{ is } \mathsf{P}\ \}$$

Remark 6.7. Here's a little game to play with friends. Look around the room, and pick a set of items (but don't tell anybody which items you picked). Then come up with a description P that you think picks out just those items. Then say, "I've selected the set containing every x such that x is P" (for example, "I've selected the set containing every x such that x is made of metal"). Have your friends then look around the room and find all the items that satisfy your description. You may discover that you need to make your description more precise!

where you can replace P with any description or recipe that tells your reader how to select the objects that go in the set.

For example, I can use set-builder notation to specify all the positive even numbers, like this:

$$\{\ x \mid x \text{ is a positive even number }\ \}$$

As another example, consider the set specified by this set-builder recipe:

$$\{\ x \mid 0 < x < 10\}$$

Read this aloud like so: "this is the set containing every x which is such that 0 is less than x and x is less than 10."

What do you think the members of this set are? Can you write it out in set-roster notation? It is this:

$$\{1, 2, 3, ..., 9\}$$

Notice that this set includes numbers from 1 to 9, but not 0 or 10. Why not? Because the recipe used the less-than symbol, not less-than-or-equal symbols.

To include 0 and 10, we would write this:

$$\{x \mid 0 \leq x \leq 10\}$$

Read this aloud like so: "this is the set containing every x which is such that 0 is less-than or equal-to x, and x is less-than or equal-to 10." And this set would include 0 and 10, as well as every number in between.

Terminology. To specify a set using **set-builder notation**, write an expression with this shape:

$$\{\ x \mid x \text{ is P }\ \}$$

and replace P with a description or recipe that uniquely identifies each x in your set.

6.6 Summary

IN THIS CHAPTER, we learned that a set is just a well-defined collection of objects. Then we learned the following:

- The objects that are **members** of a set are called the **elements** of the set (and sometimes we will just call them the **points** in a set).
- We can specify a set with **set-roster notation**, by listing out its elements between curly braces, e.g., $\{1, 2, 3\}$.
- We can specify a set with **set-builder notation**, by giving a recipe that tells the reader how to build the set themselves, e.g., $\{x | 0 < x < 10\}$.
- We name sets with **italicized uppercase letters**, e.g., A, C, and so on. We name elements with italicized lowercase letters, e.g., a, b, and so on.
- To assert that an element b is a **member** of a set C, we can use the $\in$ symbol like so: $b \in C$.
- To assert that an element b is **not a member** of a set C, we can use the $\in$ symbol with a slash through it, like so: $b \notin C$.

[7]

THE COMPOSITION OF SETS

IN THE LAST CHAPTER, we learned that a set is a collection of objects. In this chapter, we will look at the composition of sets. By looking at the elements that make up a set, we can compare sets, and also identify them.

7.1 Subsets

THE ELEMENTS OF ONE SET can be contained in another set. For example, consider these two sets:

$$C = \{1, 2, 3\} \qquad D = \{1, 2, 3, 4, 5\}$$

Every element of C is also in D. When this happens, we say that C is a **subset** of D, and D is a **superset** of C. We

Terminology. If the elements of one set are contained in another set, the contained set is called the **subset** and the containing set is called the **superset**.

symbolize it like this:

$$C \subseteq D$$

If we want to say that C is *not* a subset of D, we write this:

$$C \nsubseteq D$$

Given any sets A and B, to determine if A is a subset of B, we must check each element in A, and see if it is also in B. If we do this, and we find that every element in A is also in B, then we conclude that A is a subset of B. Let us write this down as a definition.

Notation 7.1. So A is a subset of B when: for every object x, if $x \in A$, then $x \in B$.

Definition 7.1 (Subsets)**.** For any two sets A and B, we will say that A is a **subset** of B exactly when every member x of A is also a member of B. If A is a subset of B, we will denote it like this: $A \subseteq B$. If A is not a subset of B, we will denote it like this: $A \nsubseteq B$.

As a further example, consider these two sets:

$$E = \{a, e, f, i\} \qquad F = \{a, e, i, o, u\}$$

Is E a subset of F? That is, is the following statement true:

$$E \subseteq F$$

No, because not every element in E is also in F. In particular, $f \in E$, but $f \notin F$. Hence, E is not a subset of F:

$$E \nsubseteq F$$

Now consider these two sets:

$$G = \{10, 20, 30\} \qquad H = \{10, 20, 30\}$$

Is G a subset of H? That is, is the following statement true:

$$G \subseteq H$$

In order to qualify as a subset, every element in the first set needs to be in the other set. That is true here. Every element in G is an element of H, so G counts as a subset of H.

Remark 7.1. A set A can be a subset of another set B even if A and B are the same (i.e., have the same elements).

7.2 Proper Subsets

IF $C \subseteq D$, BUT D HAS EXTRA elements that are not in C, we say that C is a **proper subset** of D. We symbolize it like this:

$$C \subset D$$

If we want to say that C is *not* a proper subset of D, we write this:

$$C \not\subset D$$

A proper subset is strictly smaller than its superset. Its superset has more elements in it. Let's make a definition for this too.

Remark 7.2. If you like, you can think of "$\subseteq$" as an analogue of "$\leq$" (i.e., less-than or equal-to), and you can think of "$\subset$" as an analogue of "$<$" (i.e., strictly less-than).

Definition 7.2 (Proper subsets)**.** For any two sets A and B, we will say that A is a **proper subset** of B exactly when every member x of A is also a member of B, and there is at least one element y in B that is not in A. If A is a proper subset of B, we will denote it like this: $A \subset B$. If A is not a proper subset of B, we will denote it like this: $A \not\subset B$.

Remark 7.3. So A is a proper subset of B when: for every object x, if $x \in A$, then $x \in B$, but there is also at least one y such that $y \in B$ and $y \notin A$.

As an example, consider these two sets:

$$J = \{a, b, c\} \qquad K = \{a, b, c, d, e\}$$

Is J a proper subset of K? That is to say, is the following statement true:

$$J \subset K$$

The answer is yes, because every element in J is also in K, but K also has some extra elements (namely, d and e).

Now consider these two sets:

$$M = \{10, 11, 12, 13\} \qquad N = \{10, 11, 12, 13\}$$

Is M a proper subset of N? That is to say, is the following statement true:

$$M \subset N$$

No, because N does not have any extra elements beyond what M has, so M cannot be a *proper* subset of N. Hence, this is true instead:

$$M \not\subset N$$

Remark 7.4. If two sets A and B have the same elements, $A \subseteq B$, but $A \not\subset B$.

However, is M a *regular* subset of N? That is, is the following statement true:

$$M \subseteq N$$

Yes. M may not be a *proper* subset of N, but M is a *regular* subset of N, because every element of M is also in N.

7.3 Identity

TWO SETS ARE THE SAME when they have the same elements. As an example, consider these two sets:

$$M = \{10, 11, 12, 13\} \qquad N = \{10, 11, 12, 13\}$$

These two sets may have different names, but they are in fact **identical** sets, because they have the same members. To symbolize that they are identical, we write this:

$$M = N$$

If we want to say they are *not* the same set, we write this:

$$M \neq N$$

To determine if any two sets A and B are the same, we must first check that every element in A is also in B, but then we must also go the other way, and check that every element in B is also in A. Let's put this into a definition.

Definition 7.3 (Set identity)**.** For any two sets A and B, we will say that A and B are **identical** sets (or synonymously: they are the **same** set) exactly when: (1) every member of A is a member of B, and (2) every member of B is a member of A. If A and B are identical sets, we will denote it like this: $A = B$. If A and B are not identical sets, we will denote it like this: $A \neq B$.

Remark 7.5. Whether two sets are the same or not is determined by their members. If they have the same members, then they are the same set. And in that case, what we are *really* saying then is that they are *one* set (one and the same set), but they have different *names*.

Remark 7.6. So $A = B$ when: for any x, if $x \in A$ then $x \in B$, and if $x \in B$ then $x \in A$.

As a further example, consider these two sets:

$$P = \{a, 1, 3\} \qquad Q = \{a, 1, 3, b\}$$

Are these the same set? No, because Q has an element that P does not. Notice that P is a subset of Q, because every element in P is in Q. Hence, this is a true statement:

$$P \subseteq Q$$

However, not every element in Q is an element of P, so Q is not a subset of P. Hence, this is a true statement:

$$Q \not\subseteq P$$

This illustrates that two sets are identical exactly when each is a subset of the other. In fact, we could define set identity using subsets if we like. We could say that two sets are identical exactly when the first is a subset of the second, and the second is a subset of the first.

Remark 7.7. An alternate definition for set equality: $A = B$ exactly when $A \subseteq B$ and $B \subseteq A$.

7.4 Duplicates and Order

DUPLICATES AND ORDER DO NOT MATTER in sets. To see this, consider the following two sets:

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

Notice that both of these sets contain elements a, b, and c, but not in the same order, and there is a duplicated "b" in the second set. Are these the same sets?

If we apply Definition 7.3 above, it turns out that yes, they are the same. Let's confirm. Is every element from the first set present in the second set? Yes:

- As for c in the first set, we can see a c present in the second set.
- As for a in the first set, we can see an a present in the second set.
- As for b in the first set, we can see a b in the second set (in fact, we can see b twice in the second set).

Now check the other way. Is every element from the second set in the first set? Yes:

Remark 7.8. Remember: to check if two sets are equal, we must check that every element in the first set is contained in the second set, and we must check that every element in the second set is contained in the first set.

- As for a in the second set, we can see an a present in the first set too.
- As for the first b in the second set, we can see a b present in the first set.
- As for the second b in the second set, we already know there is a b in the first set, because we just checked this for the first b.
- As for c in the second set, we can see a c in the first set too.

So every element contained in the first set is also contained in the second set, and every element contained in the second set is also contained in the first set. Hence, these two sets are identical.

This illustrates that duplicates and order do not matter in sets. Whether or not there are multiple copies of an element makes no difference, and the order does not matter either.

So when we talk about sets, we aren't really interested in how *many* copies something are in the set, nor are we interested in their *order*. A set is really just a container, with some things in it.

In practice, when we write out sets with set-roster notation, we simply drop any duplicates and do not write them out. Instead of writing $\{a, b, b, c\}$, we would just write $\{a, b, c\}$.

Remark 7.9. It is useful to think of a set as just a bag of elements. The order doesn't matter. We just throw the items in the bag, unorganized. Another way to say this is that a set provides structure only to the *outside* (as a container). It provides no structure *inside* the set.

7.5 Summary

IN THIS CHAPTER, we learned about the composition of sets. In particular, we learned that:

- One set A is a **subset** of another set B just in case every element of A is also an element of B. In symbols: $A \subseteq B$.

- Even if A and B have exactly the same elements, then $A \subseteq B$, because every element in A is also an element of B.
- If A is a subset of B, but B has extra elements that A does not have, then A is a **proper subset** of B. In symbols: $A \subset B$.
- If A and B have exactly the same elements, then they are **identical**. In symbols: $A = B$.
- To determine if two sets A and B are the same, we must check both ways: we must check that every element of A is also in B, and we must check that every element of B is also in A.

[8]

THE SIZE OF SETS

HOW DO WE DETERMINE the "size" of a set? In this chapter, we will will look at how to measure the size of a set, even when sets are nested inside other sets. We will also look at an important edge case: the empty set.

8.1 Cardinality

THE SIZE OF A SET is the number of elements in it. We call this the set's **cardinality**. To denote the size of a set C, we write the name of the set with vertical bars around it:

$$|C|$$

Read that aloud as: "the cardinality of the set C."

To determine the cardinality of a set, simply count the number of elements in it. For example, consider this set:

$$C = \{a, b, c, d, e, f, g, r, t\}$$

How many elements are in this set? If we count them, we see that there are nine elements in this set. So the cardinality of this set is 9. We write that like this:

$$|C| = 9$$

You can read that aloud like so: "the cardinality of the set C is 9." Let us put the notion of cardinality into a definition.

Definition 8.1 (Cardinality)**.** For any set A, the cardinality of A is the number of elements contained in A. We will denote the cardinality of A like this: $|A|$.

Remark 8.1. Because duplicates do not matter, we do not count duplicates when we count the number of elements in a set. If "b" occurs more than once in a set, we only count it one time.

8.2 The Empty Set

WE DESCRIBED A SET AS A CONTAINER. So far, we have been talking about sets that have at least some elements in them. Let's talk about sets that have very few elements in them. A set with only one element in it has a fancy name. We call it a **singleton**. But what about a set with no elements, i.e., a container with nothing inside it?

If a set has nothing in it, we call it the **empty set**. We can symbolize it in two different ways. First, we can write curly braces, and then put nothing between them, like this:

Terminology. The **empty set** has no elements in it. It is completely empty, like an empty box.

$$\{\ \}$$

The other way is a special symbol. Mathematicians use the symbol $\varnothing$ to denote the empty set.

Since the empty set has no elements in it, what is its cardinality? It has a cardinality is zero. Hence (both of these ways of writing it are equavalent):

$$|\varnothing| = 0 \qquad\qquad |\{\}| = 0$$

8.3 Peculiar Facts About the Empty Set

There are two peculiar facts about the empty set to note. First, how many empty sets are there? According to Definition 7.3, there can only be **one** empty set.

How do we know this? Two sets are identical if they have the same elements. Well, consider these two sets:

Remark 8.2. There is only **one empty set**. Hence, it is *the* empty set.

$$A = \{\} \qquad\qquad B = \{\}$$

Are A and B identical? According to Definition 7.3, we need to check if everything contained in A is also in B, and vice versa. Is that so here? Well, yes, in the sense that there is nothing in A, and we can see that there is also nothing in B. The same goes the other way: the "nothing" that's in B is also in A. So these are the same set.

The second peculiar fact about the empty set is that it is **always a subset** of every other set. According to Definition 7.1, to determine if one set is a subset of another set, we have to check that each element in the first set is also in the second set. So, consider this:

Remark 8.3. The empty set is **always a subset** of every other set.

$$\varnothing \subseteq A$$

Is $\varnothing$ a subset of A? Well, the empty set has no elements, so there's nothing to check. All of its "nothing" is also in A.

We say it is **vacuously true** that the empty set's elements are in A, because it satisfies the required definition, but only in a vacuous way. There is nothing to actually fulfill in satisfying the definition.

This is like asking, "did you finish all your chores?" Well, if you have no chores to complete in the first place, then you are indeed "finished" with your chores, in virtue of not having any to do!

8.4 Sets in Sets

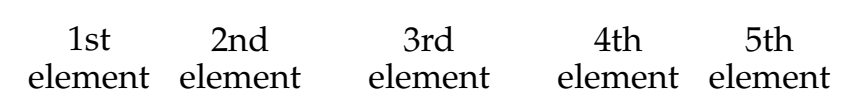
SETS CAN CONTAIN OTHER SETS. For example, consider this set:

$$\{a, b, \{1, 2, 3\}, c, d\}$$

This set has a, b, c, and d in it, but it also has $\{1, 2, 3\}$ in it, which is itself a set. How many items are in this set? That is, what is its cardinality? The answer is 5:

$$|\{a, b, \{1, 2, 3\}, c, d\}| = 5$$

Why 5? Why not 7? Well, a, b, c, and d are clearly members of this set, so that's four. But then there's only one more element, namely the set $\{1, 2, 3\}$. So that makes 5.



Remark 8.4. Be careful not to confuse **elements** and **subsets**. Take the set $\{a, b, \{1, 2, 3\}, c, d\}$. Is $\{1, 2, 3\}$ an *element* of this set, or a *subset* of it? The answer: it is an **element** of it, not a subset.

Remark 8.5. If one set is nested inside another set, when we count the number of elements in the outer set, we do not count the inner set's elements. In this example, 1, 2, and 3 are not counted as three extra elements in the outer set. Rather, the whole set $\{1, 2, 3\}$ is counted as one element of the outer set.

Now take just the 3rd element from this set, namely $\{1, 2, 3\}$. Let's unwrap it. What is *its* cardinality? The answer is 3, because there are three elements in it:

$$\{\ \underbrace{1}_{\text{1st element}}\ ,\ \underbrace{2}_{\text{2nd element}}\ ,\ \underbrace{3}_{\text{3rd element}}\ \}$$

For another example, what's the cardinality of this next set (I've written it two ways, but they both mean the same thing):

$$\{\} \qquad\qquad \varnothing$$

Think of the empty set as a box with nothing inside. When we ask for its cardinality, we are asking how many items it has inside it. The answer is zero. There are simply no items at all inside it (its an empty box).

$$\{\ \underbrace{\quad}_{\text{Nothing inside}}\ \}$$

Remark 8.6. One way to think about cardinality is to think about sets as sealed boxes that have things in them. If we unseal our box, open it up, and count how many items are inside, that's the cardinality of the set. Some of the things in our box might be smaller sealed boxes, and each such smaller box counts as one thing in our total count.

We can write that the empty set has a cardinality of zero like this (written in two ways, but both mean the same thing):

$$|\{\}| = 0 \qquad\qquad |\varnothing| = 0$$

Let us turn to another example. What is the cardinality of the following set (I've written it two ways, but both mean the same thing):

$\{\{\}\}$ $\{\varnothing\}$

This is a set that contains the empty set. It's cardinality is one, not zero. Why? Think about sets as boxes. There is an outer box, which contains an inner box (and the inner box is empty). How many things does the outer box contain? It contains one box (even though that inner box is itself empty). You can see this in the following picture (I've drawn it two ways, but again, both mean the same thing):

Remark 8.7. The set $\{\{\}\}$ is made up from an outer box that contains an inner box. The inner box is empty, but the outer box is not. The outer box has one thing in it, namely another box!

So its cardinality is one, which we can write like this (I've written it two ways, but both mean the same thing):

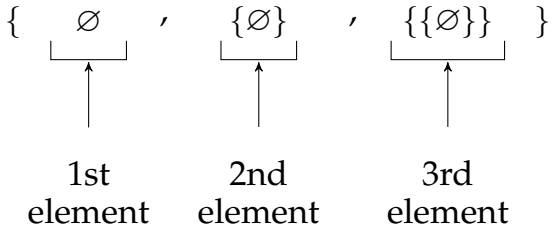
$$|\{\{\}\}| = 1 \qquad |\{\varnothing\}| = 1$$

As a final example, think about this set:

$$\{\varnothing, \{\varnothing\}, \{\{\varnothing\}\}\}$$

Think of this in terms of boxes. This box contains an empty box as its first element, then it has a box containing a box containing an empty box as its second element.

What is its cardinality? The answer is 3, because it contains three elements.



8.5 Summary

IN THIS CHAPTER, we learned how to measure the size of differents sets, including the empty set and sets with other sets nested inside of them.

- The "size" of a set is called its **cardinality**, and that is defined as the number of elements contained in the set. We denote the cardinality of a set C with two vertical bars: $|C|$.
- The **empty set** is the set with no elements. It can be written as "{ }" or "$\varnothing$."
- There is only **one empty set**, which follows from Definition 7.3.
- The empty set is a **subset of every other set**, which follows from Definition 7.1.
- The cardinality of $\{\}$ is zero, whereas the cardinality of $\{\{\}\}$ is one. That is, $|\varnothing| = 0$, but $|\{\varnothing\}| = 1$.

[9]

OPERATIONS ON SETS

ONCE WE HAVE SOME SETS at hand, we can combine them in various ways, and divide them in various ways. In this chapter, we will look at some of the operations that we can perform on sets.

9.1 Union

IF YOU HAVE TWO SETS, you can combine them into a bigger set, by taking all of the elements in the first set, and all of the elements in the second set, and then throwing all of those elements into one big new set.

When we take two sets A and B, and we join them in this way, to make a new, bigger set, we call the bigger set the

union of A and B. To denote this new, bigger set, we write this:

$$A \cup B$$

For example, take these two sets:

$$A = \{1, 2, 3\} \qquad B = \{a, b\}$$

Terminology. To form the **union** of two sets A and B, take every element from A and put it in a new set, then take every element from B and add it to the new set as well.

The union of these two sets is this:

$$A \cup B = \{1, 2, 3, a, b\}$$

Let's put this down as a definition.

Definition 9.1 (Unions)**.** For any two sets A and B, we will say that the **union** of A and B is the set that contains every element of A and every element of B. We will denote the union of A and B as $A \cup B$.

When we form a union, if there are any duplicates, we drop the duplicates. So for example, consider these two sets:

$$C = \{3, 4, 5\} \qquad D = \{4, 6, 8, 10\}$$

Note that 4 is in both sets. When we put 4 into the union, we don't need to add it to the union twice (once from C, and once from D). Duplicates are ignored in sets. Hence, the union is this:

$$C \cup D = \{3, 4, 5, 6, 8, 10\}$$

What about this one:

$$E = \{3, 4, 5\} \qquad F = \varnothing$$

What is the union of E with the empty set? Well, to form the union we take all the elements in E (which are 3, 4, and 5), and all the elements of F (of which there are none). So the union is just 3, 4, 5:

$$E \cup F = \{3, 4, 5\}$$

And that means that the union of E with the empty set is just E:

$$E \cup F = E$$

This applies to any set. The union of any set with the empty set is just the first set.

Remark 9.1. The union of any set A with the empty set is just A. That is to say, $A \cup \varnothing = A$.

9.2 Intersection

IF YOU HAVE TWO SETS A and B, you can take the elements they have in common, and put just those common elements into a new set all to their own. When we do this, we call the new set the **intersection** of A and B. To denote this new set, we write this:

$$A \cap B$$

For example, take these two sets:

$$A = \{1, 2, 3\} \qquad B = \{0, 2, 3, 5\}$$

The intersection of these two sets is this:

$$A \cap B = \{2, 3\}$$

This new set contains only two elements, because A and B only have two elements in common, namely 2 and 3. Let's put this down as a definition.

Terminology. To form the **intersection** of two sets A and B, take every element that belongs to both A and B, and put it in a new set.

Definition 9.2 (Intersection)**.** For any two sets A and B, we will say that the **intersection** of A and B is the set that contains every element which belongs to both A and B. We will denote the intersection of A and B as $A \cap B$.

When we form the intersection of two sets, if the two sets who have no elements in common, then the intersection is the empty set. For example, consider these two sets:

$$C = \{3, 4, 5\} \qquad D = \{6, 8, 10\}$$

These two sets have no elements in common, so their intersection is empty:

$$C \cap D = \varnothing$$

What about these two sets:

$$E = \{1, 2, 6, 8\} \qquad F = \varnothing$$

What is the intersection of them? What is $E \cap \varnothing$? Well, E has no elements in common with F, since F has nothing in it. So the intersection of these two sets is the empty set:

$$E \cap \varnothing = \varnothing$$

Remark 9.2. The intersection of any set A with the empty set is itself empty. That is to say, $A \cap \varnothing = \varnothing$.

This applies to any set. The intersection of any set with the empty set is empty.

9.3 Difference

IF YOU HAVE TWO SETS A and B, you can form a new set by taking every element of A, and then removing every ele-

ment that B. In your new set, you will be left with only the elements from A that are *not* in B. We call this new set the **difference** of A and B. To denote it, we write $A - B$, or $A \setminus B$.

For example, consider these two sets:

$$A = \{1, 2, 3, 4, 5\} \qquad B = \{1, 3, 5\}$$

To take the difference of A and B, take everything in A, and remove the elements that are in B. So we start with 1, 2, 3, 4, and 5, and we remove 1, 3, and 5, which leaves 2 and 4. Hence:

$$A - B = \{2, 4\}$$

Let's put this down as a definition.

Definition 9.3 (Difference)**.** For any two sets A and B, we will say that the **difference** of A and B is the set formed by taking all the elements of A and then removing any elements that are also in B. We will denote the difference of A and B like this: $A - B$.

The order of the two sets matters. Consider A and B again:

$$A = \{1, 2, 3, 4, 5\} \qquad B = \{1, 3, 5\}$$

We saw that $A - B$ is $\{2, 4\}$. But let's subtract the other way. What's $B - A$? To form the difference, we take everything from B, which is 1, 3, and 5, and then we remove any elements that are also in A. In this case, 1 is in A, so we remove it, and 3 is in A, so we remove it too, and the same goes for 5. So, the resulting set is empty:

$$B - A = \varnothing$$

Remark 9.3. If you like, you can think of $A - B$ as "A with B subtracted from it." Similarly, you can think of $A \setminus B$ as "A with B divided or separated out."

Terminology. To form the **difference** of a set A and B, take all the elements of A, and then remove any elements that are also in B. That is, "subtract" out or remove B from it.

Remark 9.4. The order of "subtracting" in set difference matters. For any two sets A and B, $A - B$ is not necessarily going to be the same as $B - A$.

What happens if the two sets have nothing in common? For example, consider these two sets:

$$C = \{1, 2, 3\} \qquad D = \{a, b\}$$

What is the difference of C and D. That is, what is $C - D$? To form the difference, we first take all the elements in C, which are 1, 2, and 3, and then we remove any elements that also happen to be in D. Well, neither a nor b occur in C, so there is just nothing to remove here. Hence, the resulting set is just 1, 2, and 3:

$$C - D = \{1, 2, 3\}$$

What is the difference of any set and the empty set? For example, consider these two sets:

$$E = \{1, 2, 3\} \qquad F = \varnothing$$

What is $E - F$? The principle here is much the same as in our last example. There is nothing in the empty set, so there is nothing to subtract from the first set E. Hence, the difference of E and F is just first set E:

$$E - F = \{1, 2, 3\} \qquad \text{i.e.} \qquad E - F = E$$

Remark 9.5. For any set A, the difference of A and the empty set is A. That is, $A - \varnothing = A$.

This applies to any set. The difference of any set and the empty set is just the first set.

What about $F - E$? That is, what if we subtract E from the empty set? To form the difference, we first take all the elements of F, which are none, and then we remove any elements that are also in E. Well, there are no elements to remove, since we're starting with nothing, so the result is the empty set. Hence:

$$F - E = \varnothing$$

Remark 9.6. For any set A, the difference of the empty set and A is empty. That is, $\varnothing - A = \varnothing$.

This applies to any set too. If you subtract any set from the empty set, the result is just the empty set.

9.4 Compliment

IF YOU HAVE A SET A and a domain of discourse, you can form a new set by taking all of the objects from the domain that are *not* in A and putting them into a set all to their own. We call this new set the **complement** of A. We denote the complement of A like this: $\overline{A}$.

Terminology. To form the **complement** of a set A, take all the elements in the domain of discourse that are not in A, and put them into a new set all to their own.

For example, suppose we have in our universe just the whole numbers 1 through 10. So, 1, 2, and so on, up to 10. Now consider this set:

$$B = \{2, 4, 6\}$$

What is the complement of B? It is the set we get by taking every object from the domain that is not in B:

$$\overline{B} = \{1, 3, 5, 7, 8, 9, 10\}$$

Let's put this into a definition:

Definition 9.4 (Complement)**.** Given a domain of discourse and a set A, we will say that the **complement** of A is the set that is formed from every object in the domain that is not in A. We will denote the complement of A as $\overline{A}$.

9.5 Summary

In this chapter, we learned about some of the ways to combine and divide sets.

- To form the **union** of any two sets A and B, take all the elements of A and all of the elements of B, and put them together into a new, bigger set. We denote the union of A and B like this: $A \cup B$.
- To form the **intersection** of any two sets A and B, take only the elements they have in common, and put them into a new set. We denote the intersection of A and B like this: $A \cap B$.
- To form the **difference** of any two sets A and B, take all the elements from A and remove any elements that are also in B. We denote the difference of A and B like this: $A - B$.
- To form the **complement** of any set A, take all of the elements from the domain of discourse that are not in A, and put them into a set all by themselves. We denote the complement of A like this: $\overline{A}$.

[10]

POWERSETS

WE CAN MAKE NEW SETS out of old sets in lots of ways. In the last chapter, we saw that we can construct the **union**, **intersection**, or **difference** of two other sets, and we can take the **complement** of a set.

In this chapter, we will talk about another kind of set that we can construct on top of any other set. It is called a **power-set**, and it is a kind of set that we will encounter again.

10.1 Powersets

SUPPOSE YOU HAVE A SET A. For example, $A = \{1, 2\}$. What are all the subsets of this set? Here they are:

$$\varnothing \qquad \{1\} \qquad \{2\} \qquad \{1,2\}$$

Just to be sure, let's confirm that each of these is indeed a subset of A:

- $\varnothing \subset A$, since the empty set is a subset of every set.
- $\{1\} \subset A$, since 1 is an element of A.
- $\{2\} \subset A$, since 2 is an element of A.
- $\{1,2\} \subset A$, since both 1 and 2 are elements of A.

Moreover, this list of subsets here is the list of all of the subsets of A that there are. We didn't miss any. We didn't forget to put one on the list.

Now, let's take all of these subsets of A, and let's put them into a set of their own:

$$\{\varnothing, \{1\}, \{2\}, \{1,2\}\}$$

This set, the set of all **subsets** of A, is called the **powerset** of A. To denote it, we write it like this:

$$\mathcal{P}(A)$$

You can read that aloud like so: "this is the powerset of A." Let's put this down in a definition:

Remark 10.1. Recall from Chapter 8 that the empty set is a subset of every set, and recall from Chapter 7 that every set is always a subset of itself.

Terminology. The **powerset** of a set A is the set of all **subsets** of A.

Definition 10.1 (Powerset)**.** For any set A, we will say that the **powerset** of A is the set of all subsets of A. We will denote the powerset of A like this: $\mathcal{P}(A)$.

Example 10.1. Consider this set: $B = \{1,2,3\}$. What is its powerset? First, we need to list out all the subsets. We begin with the empty set:

$$\varnothing$$

Then we add every singleton:

$$\varnothing \qquad \{1\} \qquad \{2\} \qquad \{3\}$$

Next, we have all of the subsets of two elements:

$$\{1,2\} \qquad \{2,3\} \qquad \{1,3\}$$

And finally, we have the entire set itself:

$$\{1,2,3\}$$

So, we collect all of these subsets together, to form the powerset of B:

$$\mathcal{P}(B) = \{\varnothing, \{1\}, \{2\}, \{3\}, \\ \{1,2\}, \{2,3\}, \{1,3\}, \{1,2,3\}\}$$

10.2 The Cardinality of Powersets

How many subsets are there for any given set? In other words, what is the cardinality of a powerset? If n is the cardinality of a set, then there will always be 2 to the power of n subsets of it.

We can write this out as an equation. We can say that the cardinality of the powerset of A is 2 raised to the power of A's cardinality:

$$|\mathcal{P}(A)| = 2^{|A|}$$

Remark 10.2. To calculate $2^{|A|}$, first calculate $|A|$ (i.e., the cardinality of A). That will give you some number n. Then calculate 2^n. For example, if $A = \{1,2,3\}$, then we first calculate $|A|$, which is 3. Then we calculate 2^3, which is 8. So $|\mathcal{P}(A)|$, i.e., the cardinality of the powerset of A, is 8.

So, if we have a set with 2 elements in it, then the cardinality of its powerset is 4, because $2^2 = 4$. So, in other words, if a set has two elements in it, then it has 4 subsets.

Similarly, if we have a set of 3 elements in it, then the cardinality of its powerset is 8, because $2^3 = 8$. That is to say, a set with 3 elements in it has 8 subsets. A set with 4 items in it has $2^4 = 16$ items, and so on. The number of subsets grows very quickly, as the original set gets bigger and bigger. For instance, a set with only 10 items in it has 1,024 subsets.

Notation 10.1. Sometimes people write "$\mathcal{P}(A)$" as "2^A," because a set with n elements in it always has 2^n subsets.

The powerset of A is sometimes written like this:

$$2^A$$

This is due to the fact that for any given set A that contains n elements, its powerset has 2^n subsets.

10.3 The Structure of Powersets

A powerset is the set of all subsets of a set. Since it includes *all* subsets, it includes the smallest subset (which is the empty set), the biggest subset (which is the entire set), and everything in between.

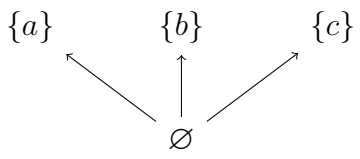
Remark 10.3. Powersets are built up from successively more complex subsets. We start with the empty set, and from there, we build up more and more complex subsets, until we reach the full set.

We can build up the powerset by starting with the empty set, and then building up more and more complex subsets on top of it. As an example, let's take the set $C = \{a, b, c\}$. Let's draw a picture that shows how the subsets are built up. First, we'll draw the empty set:

$\varnothing$

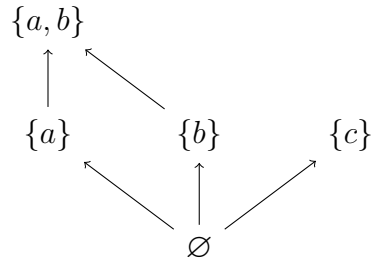
Next, above that, let's draw all of the singleton sets:

Notice that the empty set is a subset of each of these singletons (because the empty set is a subset of every set). To represent this, let's draw an arrow from the empty set up to each of the singletons.

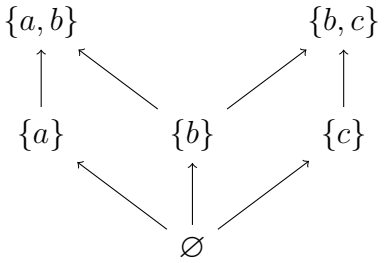


Remark 10.4. The arrows indicate that one set is a subset of another set. E.g., the empty set is a subset of $\{a\}$, it is a subset of $\{b\}$, and it is a subset of $\{c\}$.

Next, above the singletons, let's draw the subsets that have two elements in them. First, there's $\{a, b\}$, which is "composed," so to speak, of $\{a\}$ and $\{b\}$:

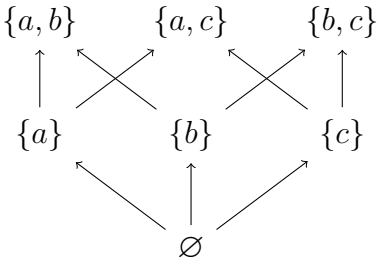


Then there's $\{b, c\}$, which is "composed" of $\{b\}$ and $\{c\}$:

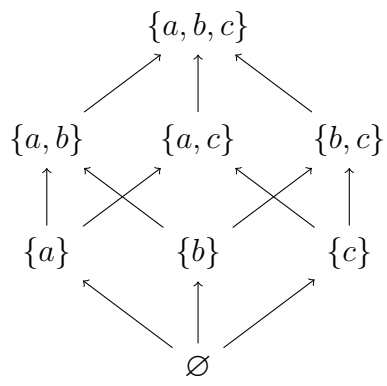


Remark 10.5. The set $\{a\}$ and the set $\{b\}$ are each a subset of $\{a, b\}$.

And there's also $\{a, c\}$, which is "composed" of $\{a\}$ and $\{c\}$:



Finally, at the top of our drawing, let's add the full set, since it's "composed," so to speak, of all the other subsets:



Notice that the arrows always indicate a subset relation. At the lowest level, the empty set is a subset of each of the singletons above it. Then, one level up, each of the singletons is a subset of the sets above them that they are connected to with an arrow. And then finally, at the top level, there is the full set, which has each of the sets below it as a subset.

This shows us that powersets have a kind of **lattice** structure. They are built up successively, starting from the empty set, and adding more and more items into the subsets at each level, until we reach the top.

Remark 10.7. When we organize the elements of a powerset into those that are subsets of others, we can see that powersets have a **lattice** structure.

10.4 Summary

IN THIS CHAPTER, we learned about powersets.

- The powerset of a set A is the set of all subsets of A.
- Powersets are built up in a lattice of progressively more complex subsets, with the simplest subset (the empty set) at the bottom, and the fullest subset (the entire set) at the top.

[11]

PRODUCTS

WE LEARNED in the last chapter how to build the **powerset** of any set. The powerset is the set of all subsets of the set in question.

In this chapter, we will talk about a further (but very important) way to build a special kind of set out of other sets. This particular kind of set is called a **product**.

11.1 Grids

TAKE ANY TWO SETS A AND B. We can use these two sets to lay out a grid. For example, consider these two sets:

$$A = \{1, 2\} \qquad B = \{a, b\}$$

Let's build a grid from these two sets. To do that, let's put all the items from A down the left side, and let's put all the items from B across the top:

	a	b
1		
2		

This gives us a simple 2 $\times$ 2 grid (rows $\times$ columns), which has 4 squares. We can navigate to any one of the four squares by specifying a row (an item from A) and a column (an item from B). For example, we can put an "X" at row 2, column b:

	a	b
1		
2		X

If we think of "(row, column)" as a **coordinate** on our grid, then we can say that our "X" is located at the coordinate $(2, b)$. This, of course, is very convenient, because it lets us give each square a name. Let's write the coordinate in each square, just so we can see all of their names:

	a	b
1	$(1, a)$	$(1, b)$
2	$(2, a)$	$(2, b)$

If we switch A and B, so that A is the columns and B is the rows, we get this:

	1	2
a	$(a, 1)$	$(a, 2)$
b	$(b, 1)$	$(b, 2)$

We say the above grid is a 2 $\times$ 2 grid, which we read out loud as "a two by two grid." What we mean by this is that

Remark 11.1. By convention, when we build grids, we make the rows grow downwards. But there is no reason we could not let the rows grow upwards, like this:

2		
1		
	a	b

Terminology. A **coordinate** is a pair of the form (x, y), with x replaced by a row, and y replaced by a column.

this is a grid with a certain number of rows, and a certain number of columns (2 rows and 2 columns).

We get the grid by **crossing** the rows with the columns, so to speak. That is to say, we extend or **project** the rows outwards and the columns downwards, so that they cross each other. Each cell in the grid is basically just one point where the two projected rows/columns cross.

So, when we use the "times" symbol in "2×2," we don't exactly mean multiplication from arithmetic. We mean something more like **cross**. And notice that the "times" symbol itself is exactly two crossed lines!

However, we can multiply the number of rows by the number of columns, and that will tell us the total number of cells in our grid. In this case, 2 (rows) times 2 (columns) gives us 4 (cells):

$$\mathsf{rows} \times \mathsf{columns} = \mathsf{cells}$$

So we can use the "times" symbol here to mean plain-old multiplication too, when we want to talk about the number of *cells* (or *points*) in the grid. But, usually, when we are talking about grids, we use the "times" symbol to mean "cross," so we can read "2×2" as "2 by 2," or "2 crossed with 2," or even more simply, just "2 cross 2."

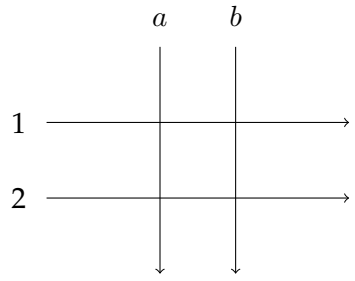
A grid is built from $\mathsf{rows} \times \mathsf{columns}$. To **name a grid**, we can replace "rows" and "$\mathsf{columns}$" with the sets we use to build them. So, for instance, we can say

$$A \times B$$

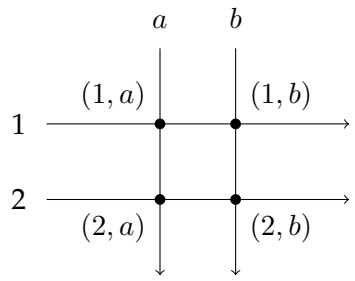
(read that as "A cross B") to describe the grid we get when we use A for the rows and B for the columns. Likewise, we can say

$$B \times A$$

Remark 11.2. We often draw grids like a spreadsheet table, with the cells as boxes that you can fill in with your own writing. But we could just as easily think of **projecting** the items in A and B outwards, as **lines**:



Then, we can see that these line projections make a grid from the crossed lines, and the cells are actually all of the **points** where the lines cross:



(read that as "B cross A") to describe the grid we get when we use B for the rows and A for the columns.

Example 11.1. Consider the following two sets (which are different in size):

$$C = \{1, 2\} \qquad\qquad D = \{a, b, c, d\}$$

Here is the grid of $C \times D$:

	a	b	c	d
1	$(1, a)$	$(1, b)$	$(1, c)$	$(1, d)$
2	$(2, a)$	$(2, b)$	$(2, c)$	$(2, d)$

This is a grid with 2 rows and 4 columns. How many cells are there in total? There are 8, because 2×4 is 8.

Example 11.2. If we reverse the rows and columns, we get a different grid, namely $D \times C$:

	1	2
a	$(a, 1)$	$(a, 2)$
b	$(b, 1)$	$(b, 2)$
c	$(c, 1)$	$(c, 2)$
d	$(d, 1)$	$(d, 2)$

Remark 11.3. When we build a grid from two sets C and D, the **order matters**: $C \times D$ is not the same grid as $D \times C$.

Notice how this is also a grid with 8 cells in it, but it is obviously quite different from the $C \times D$ grid.

11.2 Products

TAKE TWO SETS A AND B, and build their grid. Next, take all the points in that grid (all the coordinate-pairs), and put

them into a set. We call this set the **product** of A and B (synonymously, we can call it the **cross product** or the **Cartesian product** of A and B, but these are all just synonyms). As an example, consider the sets A and B from above:

$$A = \{1, 2\} \qquad B = \{a, b\}$$

If we build the grid, and take all of the points (i.e., all of the coordinate pairs), we get these:

$$\begin{array}{ll} (1, a) & (1, b) \\ (2, a) & (2, b) \end{array}$$

Next, let's put all of these pairs into a set. To do that, we simply list them out, and wrap curly braces around them (set-roster notation):

$$\{(1, a), (2, a), (1, b), (2, b)\}$$

This is the **product** of A and B, and we name it the same as its grid:

$$A \times B$$

Hence, we can specify the product of A and B more fully, like this:

$$A \times B = \{(1, a), (1, b), (2, a), (2, b)\}$$

The product is really just a grid, but stripped down to its bare essentials. We get rid of the pictures and geometrical layout, and all we're left with is the set of coordinate-points. That's the product.

Indeed, if we forget about the grid, and just look at what we see on the page before us here, we can see the following:

Terminology. The **product** is also called the **cross product** or the **Cartesian product**.

Notation 11.1. The product of sets A and B is written like this: $A \times B$. Read this out loud as "the product of A and B." You can also read it like this: "A crossed with B," or even just "A cross B."

Remark 11.4. Each pair has two slots: a first slot, and a second slot. Elements from A go in the first slot, and elements from B go in the second slot.

- Each element of the set is a pair. A pair is a sequence of two items: so there is a slot for a first item, and then a slot for a second item.
- The first slot is always filled with an element from A, the second with an element from B.
- Every possible pairing of an element from A and an element from B are in this set. None have been left out.

With all of that said, we have enough to put together a definition.

Definition 11.1 (Products)**.** For any sets A and B, we will say that the **product** of A and B is the set of every pair (x, y) that can be constructed by replacing x with an element from A and by replacing y with an element of B. To denote the product of A and B, we will write this: $A \times B$.

Remark 11.5. No possible pair is left out of the product of A and B. Every combination that we can get by taking something from A first and something from B second is included in the product. This makes sense if we remember that the product is the set of *all* coordinate-pairs from its grid.

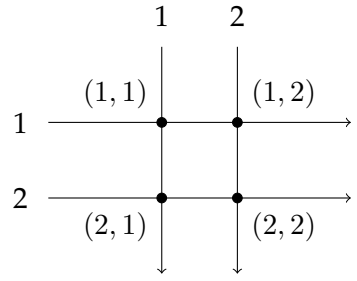
Example 11.3. What about $A \times A$? We build this product just as the others. We make the pairs by taking one element from the first set (in this case A), and another element from the second set (which in this case is A again). Here are all the possible pairs:

$(1, 1)$	$(1, 2)$
$(2, 1)$	$(2, 2)$

Then we put all of these pairs into a set, to get our product:

$$A \times A = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$$

Remark 11.6. It may help to visualize the coordinates of the $A \times A$ grid:



11.3 The Cardinality of Products

HOW BIG IS A PRODUCT? That is to say, what is the cardinality of the product of A and B?

$$|A \times B| = \text{ ??}$$

We know that the number of coordinate pairs in a grid is:

$$\text{rows} \times \text{columns} = \text{coordinates}$$

How do we know the number of rows? It's the number of elements in A. In other words, it's the *cardinality* of A. Likewise for the number of columns: it's the number of elements in B, i.e., the *cardinality* of B. Hence, we can calculate the cardinality of any product by multiplying the cardinality of the two sets together:

$$|A \times B| = |A| \times |B|$$

Remark 11.7. Recall that the cardinality of a set is the number of items in it. The cardinality of the sets in a grid tell us how many rows or columns there are in the grid, because when we build a grid, we put each element from the first set at the head of its own row, and we put each element from the second set at the head of its own column. Hence, there's going to be as many rows as there are elements in the first set, and there are going to be as many columns as there are elements in the second set.

Example 11.4. Consider these sets:

$$C = \{1, 2\} \qquad D = \{a, b, c\}$$

What is the cardinality of $C \times D$? The cardinality of C is 2, and the cardinality of D is 3, and 2 times 3 is 6:

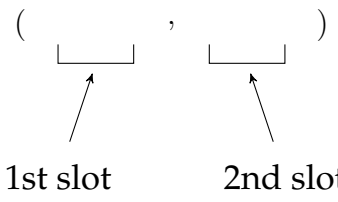
$$|C \times D| = |C| \times |D| \qquad \text{i.e.,} \qquad 6 = 2 \times 3$$

And indeed, that makes sense. If we look at the roster of $C \times D$ pairs, we can see that there are exactly six pairs:

$$C \times D = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$$

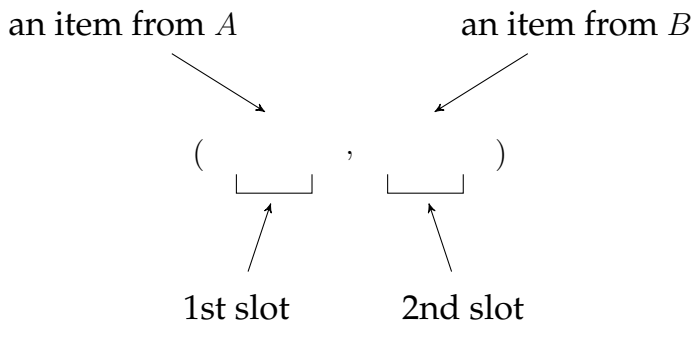
11.4 The Structure of Pairs

LET'S LOOK AT THE STRUCTURE of the pairs that make up the product of any two sets A and B. Each pair is like a container that has two slots, one after the other. Something like this:



Remark 11.8. A pair is container with two slots, where the order matters. The first slot comes first, and the second slot comes second.

The first slot can only be filled with items taken the first set (i.e., A), and the second slot can only be filled with items taken from the second set of the product (i.e., B).



Remark 11.9. Each slot can draw its values only from a specific **source**. E.g., the first slot can only draw from A, and the second only from B. We can say that each slot can only be filled by values of a certain **type**. Hence, the first slot can only be filled with values of type A, and the second slot with values of type B.

This highlights how the order matters, and how each slot has a specific "source" that it can draw elements from.

In this regard, pairs have quite a bit **more structure** than a set. Remember that a set has no internal structure. A set is just a bag of items. We don't care about the order they come in, and we don't care about duplicates.

A pair is not like this. A pair does **have an order**, and the **source** of values for each slot matters too! A pair for $A \times B$ is

different from one from $B \times A$, $A \times A$, or $B \times B$. So a pair is not just an unstructured *set* of two items. It is a highly structured container in its own right.

11.5 Tuples

WE CAN MAKE PRODUCTS out of more than just two sets. We can make the cross product of three sets, or four sets, and so on. The rules for building them are the same, except that we fill in not just two slots, but rather three slots, four slots, and so on.

As an example, let's make a cross product of three sets. Consider these three sets:

$$A = \{1, 2\} \qquad B = \{a, b\} \qquad C = \{r, s\}$$

The cross product of $A \times B \times C$ is built by forming all triplets with an element from A first, an element from B second, and an element from C third. Here are all the triplets:

$$\begin{array}{cccc} (1,a,r) & (1,b,r) & (2,a,r) & (2,b,r) \\ (1,a,s) & (1,b,s) & (2,a,s) & (2,b,s) \end{array}$$

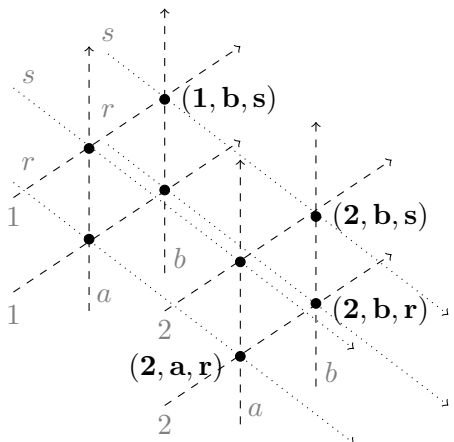
And if we put them into a set, we then have our product $A \times B \times C$:

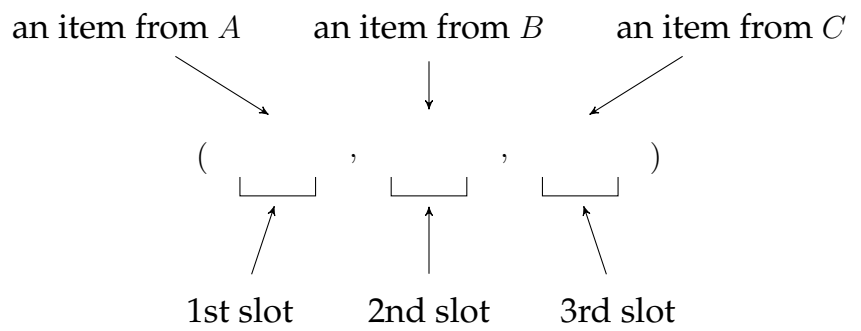
$$A \times B \times C = \{(1,a,r), (1,b,r), (2,a,r), (2,b,r), \\ (1,a,s), (1,b,s), (2,a,s), (2,b,s)\}$$

Note the structure of each of these triplets:

Terminology. A **triplet** (synonym: **triple**) is just like a pair, but it has three slots instead of two.

Remark 11.10. Think of making a 3-dimensional grid with A, B, and C. If we think about projecting these sets (one in each of 3 dimensions), so that they make lines that cross, there will be 8 points where the lines cross. Here is a picture, with a few of the points labeled. See if you can label the rest:





So this is exactly like what we have in pairs, except there are three slots, rather than two slots.

In the same way, we could build a product of four sets. For example, we could build $A \times B \times C \times D$, and that would be a set of quartets. Each quartet would be a sequence of four slots, with a value in the first slot taken from A, a value in the second slot taken from B, a value in the third slot taken from C, and a value in the fourth slot taken from D.

More generally, these sequences of slots are called **tuples**. If there are two slots, we call it a **2-tuple**. If there are three slots, we call it a **3-tuple**. If there are four slots, we call it a **4-tuple**, and so on for any **n-tuple**, where n is any positive whole number.

Remark 11.11. A product of four sets is the set of all coordinates of a four dimensional grid made from the sets in question. But of course, it is really hard to imagine grids beyond three-dimensions, so it is much easier to think of the product of four sets simply as the set of all possible quartets made from the sets in question.

Terminology. An **n-tuple** is a sequence of n slots, where n is a positive whole number. We can also call 2-tuples "**pairs**," 3-tuples "**triplets**" (or **triples**), 4-tuples "**quartets**", 5-tuples "**quintets**" (or quintuples), and so on.

11.6 Summary

IN THIS CHAPTER, we learned about how to cross any number of sets, in order to construct the set of all coordinate-pairs that can be built from them. We call these sets of coordinates the **product** of the sets we crossed.

When we construct products, we are basically just constructing grids (in 2, 3, or more dimensions). However, we drop the pictures and lines and other inessential information,

and we basically just strip the information down to its bare essentials, namely the set of possible coordinates.

- The **grid** of two sets $A \times B$ is the grid you get by making all the elements of A into rows and all the elements of B into columns. The **coordinates** of such a grid are all $(\mathsf{row}, \mathsf{column})$ pairs.
- The **product** of two sets $A \times B$ is the set of all coordinate-pairs of the grid we would get from A and B.
- The **pairs** of a product $A \times B$ are each a **sequence** with **two slots**. The first slot can only be filled with an element from A, and the second slot can only be filled with an element from B.
- We can form the **product of n sets**, where n is a positive whole number. The members of such a product are called **n-tuples**, which are sequences of n slots.

[12]

FURTHER READING

To pursue sets and set theory further, the following list may offer some helpful starting points.

- Stewart and Tall (2015, ch. 3) provides a good but introductory-level discussion to the basic concept of set theory.
- Steinhart (2018, ch. 1) also provides an excellent introduction for the beginner to the topic of sets and set theory.
- Stewart (1995, ch. 4) offers a discussion of set theory for non-experts.
- Flegg (1974, ch. 14) provides a conceptual discussion of sets and set theory.
- Halmos (2017) offers a non-technical discussion of the classic topics in set theory.

- Cummings (2020, ch. 3) presents a slow, thorough discussion of sets and how to construct proofs about them.
- Velleman (2019, chs. 1–2) discusses logic in these chapters, but in the midst of all that, he provides an excellent introduction to the concept of sets and proofs about them.
- Zach (2019, ch. 1) provides an introductory-level discussion of sets, with proofs and some helpful discussion.
- Warner (2019, chs. 1–2) covers sets in a thorough way, with worked out proofs for all details.
- Madden and Aubrey (2017, ch. 9) provides a discussion of sets and how to build proofs about them.
- Pinter (2014, ch. 1) offers a simple discussion of the basic theorems and proofs for set theory.
- Jongsma (2019, chs. 4–5) provides a rigorous introduction to many of the higher-level topics of set theory.
- Enderton (1977) is a classic textbook for set theory. If you find set theory deeply fascinating, turn to this after you are comfortable reading proofs.
- Smullyan and Fitting (2010) is another textbook that thoroughly covers most of the classic topics in modern set theory. Reach for this one after you are comfortable reading proofs, and if you are interested in the many deep and interesting questions that can be found in set theory.

Part [3]

FUNCTIONS

Once we have organized all of the objects we care about into sets, we can then start building associations or "mappings" between them. A mapping between two sets simply maps the items in the first set to items in the second set. We can put together as many mappings as suit our purposes. Mathematicians call a mapping a **function** (or synonymously, a **map**), and this basic idea is absolutely fundamental to almost every branch of math.

In Part 3, we discuss **functions** (or **maps**). We discuss what they are, how they are defined, and many of their properties. We also see how they help us define one of the most fundamental concepts in math: **isomorphisms**.

[13]

MAPPINGS

WE VERY OFTEN make mappings from one set of things to another set of things. For example, a phone directory maps people to phone numbers, whereas a product catalogue maps serial numbers to prices.

In this chapter, we will look at a very common way to present a mapping, called **lookup tables**. We will also learn about another way to present the same information, called **association lists**.

Ponder. What kinds of mappings do you use to organize things, remember things, or even to understand things?

13.1 Lookup Tables

IN ESSENCE, a **lookup table** is a table that indexes some data with lookup keys. A key could be a serial number, a last

name, or whatever, so long as it is unique. A key is just a unique identifier, that we can use to pick out the piece of data we are looking for in the table.

Terminology. A **lookup table** indexes information by keys. It makes it easy for us to lookup a particular piece of data by its key. A **key** is just a unique label that lets us identify the piece of data we want to lookup.

As an example of a lookup table, consider the following office phone directory:

Employee Name	**Phone Number**
Akimbe, T.	671-476-3455
Carleno, S.	671-476-3456
Haslinger, T.	671-476-3457
McRogers, K.	671-476-3458
...	...

If we want to lookup somebody's phone number using this table, all we need to know is their last name. For example, suppose we want to find T. Haslinger's office phone number. To do that, we find "Haslinger, T." in the **Employee Name** column, then we look over to the phone number associated with T. Haslinger, which is listed in the right column.

Terminology. The **data** of a lookup table is the information we can lookup. The **keys** are the labels we attach to each piece of data.

In essence, what we have here is just a list of phone numbers, keyed by names. The **keys** of this lookup table are *names*, and the **data** we can lookup are *phone numbers*.

13.2 Meta Information

Let us establish some terminology and notation that will help us describe the **meta information** of the table.

Remark 13.1. The **meta information** of a table is the **name** we use to refer to it, along with information about what **kind of data** it contains, and what **kind of keys** it uses. (Is it a table of phone numbers? Home addresses? What sort of data is in it? And what are the keys?)

Names. Let us say that we can give a **name** to a lookup table. That way, we can refer to the table by name, so that other people will know which table we are talking about (if we are talking about more than one table at a time). In this case, let us give this lookup table the name "directory."

Domain and codomain. Let us call the *keys* the **domain** of the lookup table, and let us call the *data* the **codomain** of the table.

In other words, the *domain* is the information in the left column, i.e., it is the set of unique keys that we use to lookup values, and the *codomain* is the data in the right column, i.e., it is the set of values that we lookup.

Terminology. The **domain** of a lookup table is the set of keys (the left column), and the **codomain** of the table is the set of data values its keys are associated with (the right column).

The type of table. Let us write the table's name, domain, and codomain like this:

$$\mathsf{directory} : \mathsf{names} \to \mathsf{phone\ numbers}$$

Read that aloud like this: "the lookup table called directory maps *names* to *phone numbers*." This is a very compact way of describing what kind of lookup table we are dealing with. It tells us the name of the table, and it tells us what the domain is (names) and what the codomain is (phone numbers). This gives us, so to speak, the *signature* of the lookup table.

In general, we will write the **signature** of a lookup table with this format:

$$\mathsf{mapping} : \mathsf{domain} \to \mathsf{codomain}$$

where:

- mapping is replaced with the name of the table (e.g., "directory," "product catalogue," or whatever name we have given the table).
- domain is replaced with the type of keys the table uses (e.g., "names," but it could be "serial numbers," etc).
- codomain is replaced with the type of data we can lookup in the table (e.g., "phone numbers," "prices," etc).

Example 13.1. Consider this table:

$$\mathsf{players} : \mathsf{names} \to \mathsf{position}$$

This describes a table that maps the players on a sports team to the positions they play. Their names go in the left column, and the position they play goes in the right column.

Terminology. We denote the **signature** of a lookup table like this:

$$\mathsf{mapping} : \mathsf{domain} \to \mathsf{codomain}$$

where mapping is the name of the table, domain is the type of key, and codomain is the type of data you can lookup.

13.3 Values in the Table

Now let us establish some terminology and notation that will help us describe the **data** in the table more concisely. Suppose we want to say something like this:

> According to this directory, the phone number assigned to T. Haslinger is 671-476-3457.

That takes quite a few words to write out. We can definitely shorten this. Perhaps we could write something like this:

$$\text{In this directory, T. Haslinger} = \text{671-476-3457}$$

That is certainly shorter, but we can make it even more concise. Let's write it like this:

$$\text{directory}(\text{T. Haslinger}) = \text{671-476-3457}$$

This tells us that this "directory" gives the number "617-476-3457" for the name "T. Haslinger." Here is another example:

$$\text{directory}(\text{T. Akimbe}) = \text{671-476-3455}$$

That tells us that this "directory" gives the number "617-475-3455" for the name "T. Akimbe."

Now consider the following expression. Which particular value does it refer to?

$$\text{directory}(\text{K. McRogers})$$

The answer is: "617-476-3458." After all, if we use this "directory" to lookup the phone number associated with the name "K. McRogers," we will find that the number it gives us is "617-476-3458."

Notation 13.1. To assert that a particular **key** is associated with a particular **value** in a lookup table, use an expression with this shape:

$$\text{mapping}(\text{key}) = \text{value}$$

where "mapping" is the name of a lookup table, "key" is one of the keys, and "value" is the piece of data associated with that particular key.

In fact, when we write down "directory(K. McRogers)," we can just think of that as a synonym for the phone number "617-476-3458." So, if we want to refer to K. McRogers' phone number, we can either write out the phone number itself:

617-476-3458

or we can write out this:

directory(K. McRogers)

because anybody can use the table to go and lookup what the phone number for K. McRogers is for themselves.

Example 13.2. This expression:

directory(S. Carleno)

is a synonym for this:

617-476-3456

It is just another way of referring to Carleno's phone number.

13.4 Association Lists

WE CAN ALSO THINK OF A LOOKUP TABLE as an **association list**, because it is really just a list of associated pairs: it associates phone numbers with names. Let's write out our directory as a list of associated pairs:

Terminology. An **association list** is a list of associated pairs. In our example, it is a list of (name, phone number) pairs.

$$\begin{aligned}\text{directory} = {} & (\text{T. Akimbe}, 671\text{-}476\text{-}3455),\\ & (\text{S. Carleno}, 671\text{-}476\text{-}3456),\\ & (\text{T. Haslinger}, 671\text{-}476\text{-}3457),\\ & (\text{K. McRogers}, 671\text{-}476\text{-}3458),\\ & \ldots\end{aligned}$$

Here we can see that we have a list of pairs. In each pair, the first item is a name (the "key"), and the second item is a phone number (the "data"). Even though we have changed the format, the essential contents of the lookup table are present in this association list.

Also, notice that this association list has the same **meta information** as the lookup table. It has the same name, the type of keys are the same (employee names), and the type of data is the same too (phone numbers). Hence, we can describe this association list with the same **signature** that we used to describe the lookup table:

Remark 13.2. We can denote the signature of an association list the same way that we can denote the signature of a lookup table: mapping : domain → codomain.

$$\text{directory} : \text{names} \to \text{phone numbers}$$

Notice also that we can refer to **particular values** in this association list just as we did with the lookup table. For example, we can write this:

Remark 13.3. With an association list, we can assert that a particular key is associated with a particular value the same we do with a lookup table: mapping(key) = value.

$$\text{directory}(\text{T. Haslinger}) = \text{671-476-3457}$$

And this means exactly the same as it did before: it means that in this directory, the phone number associated with the name "T. Haslinger" is "617-476-3457."

Similarly, consider this expression:

$$\text{directory}(\text{S. Carleno})$$

Remark 13.4. We can think of a **lookup table** as an **association list**, and vice versa, since they have the same contents. They are two different ways of presenting the same information.

We can read this aloud as "the phone number that this directory has for the name "S. Carleno." This expression refers to the phone number "671-476-3456," because that is the number it associates with the name "S. Carleno."

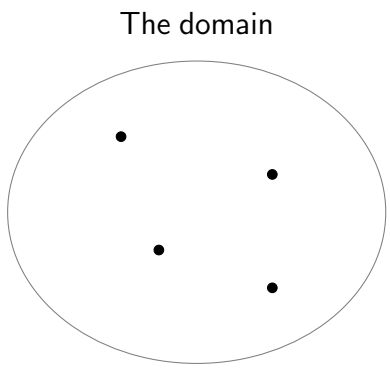
All of this makes it clear that we can think about any lookup table as an association list, in just the way we have done here. Be it a lookup table or an association list, the **essential contents are the same**. We have some data (e.g., phone numbers), and each piece of data is labeled by a unique key (e.g., employee names).

13.5 Mappings from Keys to Values

LOOKUP TABLES AND ASSOCIATION LISTS are two different ways to present the same information. Either way, they present us with a **mapping** from **keys** to **values**.

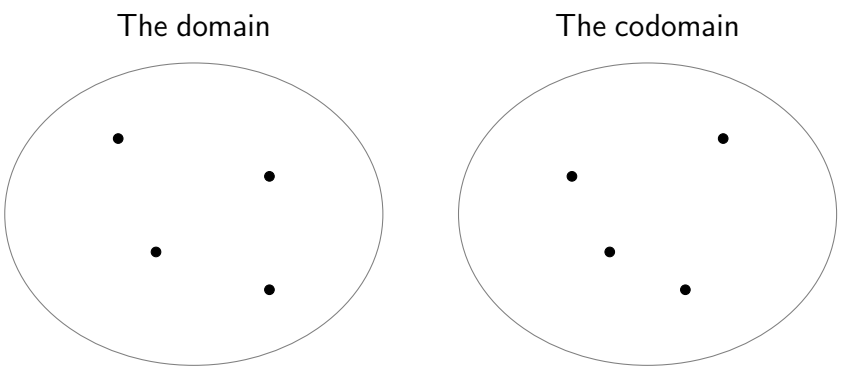
Let's picture this a bit more abstractly. Let's think about the domain as just a bag of points:

Remark 13.5. Recall that one way to think about a **set** is as a bag of items. We are pretty much doing that here. We're thinking of the domain as a bag of items (which we are representing as points).

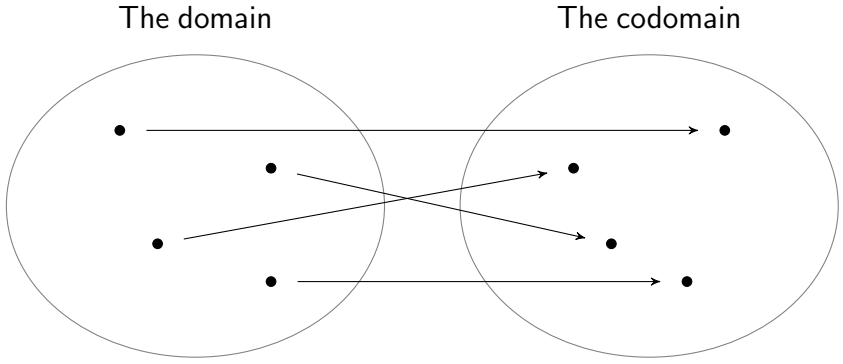


Let's think about the codomain as another bag of points:

Remark 13.6. Two separate bags of points are like two separate sets. Each is a container with its own items inside it.

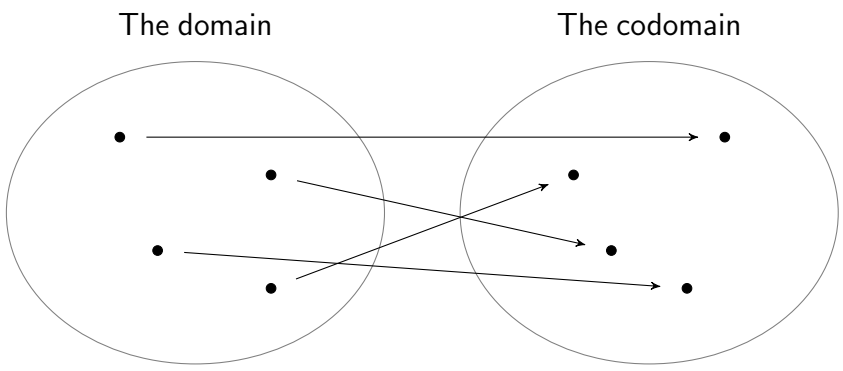


A **mapping** connects up points from the left side to points on the right side. We can draw this with arrows:



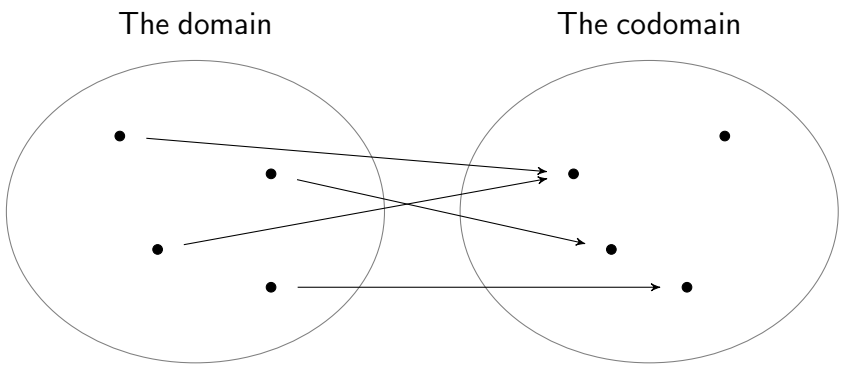
Terminology. One way to think about a **mapping** is in this pictorial way. It is the arrows which map the points on the left to the points on the right.

There are more ways to connect the points besides the one I just drew. Here is another mapping:

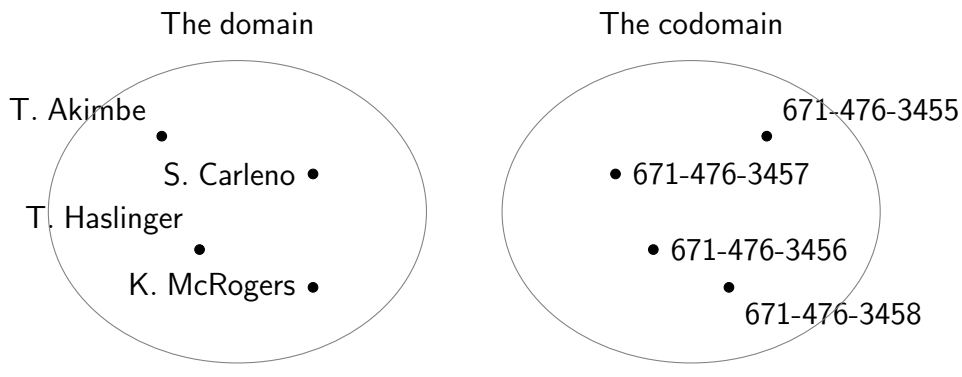


And here is yet another mapping:

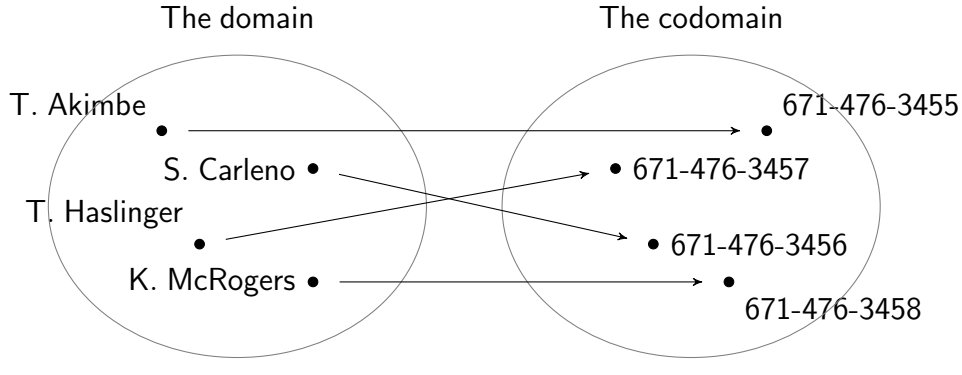
Remark 13.7. Notice that multiple points in the domain can be mapped to the same point in the codomain. And notice that some points in the codomain can have no arrows at all pointing to them.



Let's add some labels to the points, so that we can think of them as representing the names and phone numbers from our lookup table:



Once we add the labels, we can then draw our mapping:



It is easy to see that these pictures capture the same information that we have in the lookup table and association list. All of these are really just **mappings**, which map **keys** (i.e., items on the left) to **values** (i.e., items on the right).

13.6 Summary

IN THIS CHAPTER, we learned about **lookup tables** and **association lists**, which both present the same information. They index some data with unique keys, for easy lookup. In essence, they present us with a **mapping** from keys to data values.

- A **lookup table** is a table with two columns. The keys go in the left column, and the data goes in the right column.
- An **association list** is a list of pairs, with the key as the first item in the pair, and the data as the second item.
- We call the keys the **domain**, and the data the **codomain**, of the mapping.
- We can describe the **signature** of a mapping like this:

  $$\mathsf{mapping} : \mathsf{domain} \to \mathsf{codomain}$$

  where "mapping" is replaced with the name of the mapping, "domain" is replaced by the type of keys, and "codomain" is replaced by the type of data.
- We can assert that a particular key is associated with a particular value in such a mapping like this:

  $$\mathsf{mapping}(\mathsf{key}) = \mathsf{value}$$

  where "mapping" is replaced with the name of the mapping, "key" is replaced by one of the keys, and "value" is replaced with the value associated with that particular key.
- An expression with the shape "mapping(key)" is a synonym for the value associated with that key.

[14]

FUNCTIONS

IN THE LAST CHAPTER, we discussed **mappings**, which map keys to values. We initially looked at lookup tables and association lists, but we saw that both of those are really just two different ways to present the same thing. If we get a bit more abstract and think in terms of pictures, we can see that a mapping is really just a bunch of arrows that connect up the items in one bag to items in another bag.

In math, we typically call mappings **functions**, although many mathematicians sensibly just call them **maps**. Whatever we choose to call them, functions are fundamental to math. It is practically impossible to look at any branch of math without finding functions everywhere. In this chapter, we will more formally introduce the basic concepts, terminology, and notation that govern the use of functions.

Terminology. A **function** or **map** is basically just a mapping from keys to values, much like the lookup tables or association lists we discussed in the last chapter.

14.1 Terminology and Notation

A FUNCTION is a mapping, in exactly the same sense that we discussed in the last chapter. So, a function maps keys to values. But let's get a little more precise about this.

We saw that we can think of a function as a map from the items in one bag to the items in another bag, and of course the idea of a "bag of items" is just the idea of a **set**. So let us talk about this in terms of sets. Let us say that for any function, we start with two sets:

- We call the first set the **domain**.
- We call the second set the **codomain**.

Then, we can describe a function like this:

- A function maps **every element** from the domain to **one element** in the codomain.

Terminology. A **function** (or synonymously, a **map**) maps the items from one set to the items in another set. The first set is called the **domain** of the function, and the second set is called the **codomain** of the function.

In the last chapter, we named our mapping "directory." But this is a fairly long name to write out, and mathematicians like short names. When mathematicians name functions, they usually just pick the names f, g, and h. We will do this too.

Terminology. We will use short names for functions like f, g, or h.

We can describe a function's **signature** in the same way that we described the signature of a mapping in the last chapter. For example, to denote the signature of a function f that maps items in a set A to items in a set B, write this:

$$f : A \to B$$

Notation 14.1. To denote the **signature** of a function, use this format:

$$f : A \to B$$

where f is the name of the function, and A and B are sets (the domain and codomain of the function, respectively).

Read that aloud as "f maps items from A to items in B," or "f takes elements from A to elements in B," or even more concisely, "f goes from A to B."

For another example, suppose we want to describe the signature of a function g that maps elements from a set C to elements from another set D. To do that, write this:

$$g : C \to D$$

Read that aloud as "g goes from C to D," or "g takes elements from C to elements in D," or even "g maps items from C to items in D."

We can also assert that a function maps a particular key to a particular value in the same way that we did for mappings in the last chapter. Suppose we want to assert that a function f maps an item x to the item y (where x is an item in the set A and y is an item in the set B). To do that, write this:

$$f(x) = y$$

Read that aloud like this: "f maps x to y." Mathematicians also read that like this "f applied to x yields y," or more concisely, "f of x is y."

In the last chapter, we pointed out that an expression like "directory(T. Haslinger)" is just a synonym for the phone number that the directory associates with "T. Haslinger." The same applies here. Take this expression:

$$f(x)$$

Read that aloud as "f applied to x," or "f of x," or "the value that f gives for x." The expression "$f(x)$" is just a synonym for the value y that f associates with x. So we can write the value directly if we like, i.e., "y," but we can also write "$f(x)$" instead, and that means the same thing. Both expressions refer to y.

From all of this, you can see that the terminology and notation for functions is pretty much exactly the same as what we used for lookup tables and association lists in the last chapter.

Notation 14.2. To assert that a function f maps an element x to an element y, use this format:

$$f(x) = y$$

where f is the name of the function, x is an element from the domain, and y is the element in the codomain that f maps x to.

Notation 14.3. This expression

$$f(x)$$

is pronounced "f of x" or "f applied to x." It is a synonym for the value that f associates with x.

14.2 The Definition

Now that we have established the basic terminology and notation, let us put down what we have said into a proper definition for functions.

Definition 14.1 (Functions)**.** For any two sets A and B, a **function** (or **map**) f from A to B is a mapping which maps each element from A to one element from B. To denote that f maps elements from A to elements from B, we will write $f : A \to B$, and we will call this the **signature** of f. To denote the element from B that f associates with x, we can write $f(x)$. To denote that f maps $x \in A$ to $y \in B$, we will write $f(x) = y$.

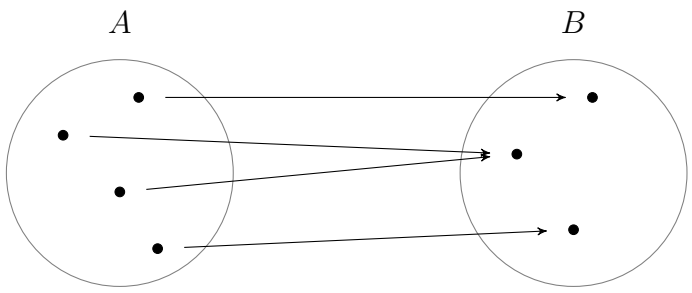
Remark 14.1. Recall that sets are like containers: they hold things inside of them.

To help visualize this, let us draw some pictures. First, let us draw two sets. Since we can think of a set as a container, let's draw each set as a circle, as if to imagine that it is a bag that can hold things inside of it. We do not care what these sets are called just yet, so let's just pick the names A and B. Here they are:

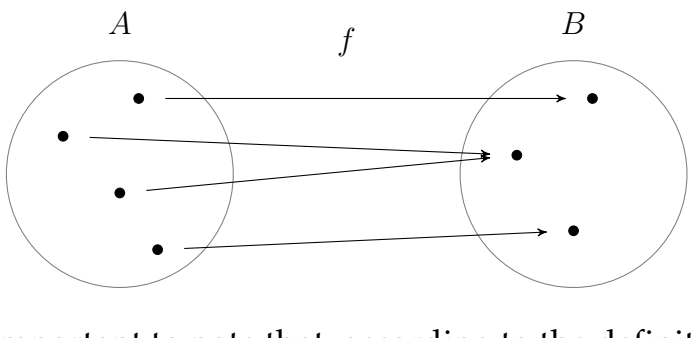
Now let's add some elements to these sets. We do not care exactly what the elements are yet, so we can just draw them

as points scattered about inside their containers. Here are some random points put into each container:

At this point, we can draw in a mapping. Here is one:



Let's give this map a name. Let's call it f. We'll write its name above the arrows:



Remark 14.2. Pictures like this show us how f maps each element from A to an element from B. We can call this the **diagram** of f.

It is important to note that, according to the definition of a funtion, a function must map **each point** in the first set to **one point** in the second set. Is that true for f? In the picture we

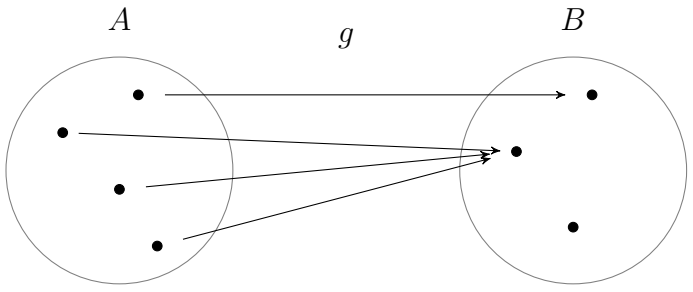
just drew, we can see that f does indeed map each element in A to one element in B.

Here is the way to check that a mapping fully maps each element in the domain to one element in the codomain. Check these two things:

- Check that there is **exactly one arrow** coming out of each point in the domain set. This tells us that the function maps each point to **one element** in the codomain. If we saw *two* arrows coming out of a single point in the domain, we would know that the mapping is trying to connect up a point in the first set to *more than one* point in the second set, and that's not allowed. A function must map each point from the domain to **exactly one** point in the codomain.
- Check that the domain has **no arrowless points**, i.e., make sure that there are no points in the codomain which have no arrows coming out of them. The definition of a function says that a function must map **each point** in the domain to a point in the codomain. So, there can't be any points in the domain that don't have an arrow coming out of them. If you find a point in the domain that has no arrow coming out of it, then this mapping is not a genuine function.

Remark 14.3. A function maps **each point** in the domain to **one point** in the codomain. So, every point in the domain will have *exactly one* arrow coming out of it. There cannot be points in the domain that have no arrows coming out of them, or more than one arrow coming out of them.

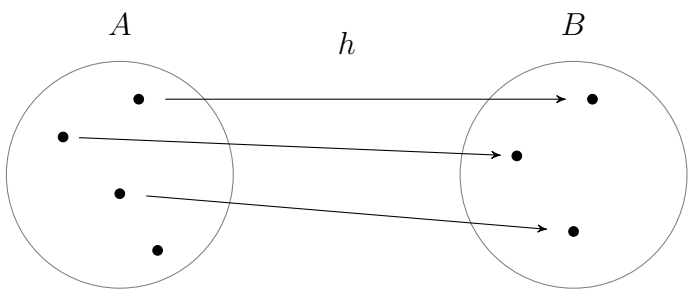
Let's look at some examples. Consider this mapping:



Is g a function? Yes, because each point in A has exactly one arrow coming out of it. Notice that three of the points in A point to the same point in B. This is allowed in a function. Nothing says that a function can't map multiple points in the domain to the same point in the codomain.

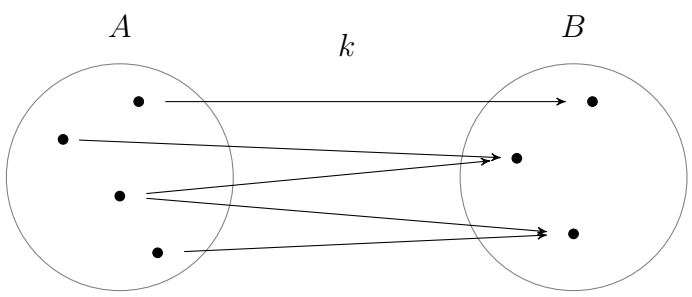
Notice also that there is a point in B that has no arrows pointing at it. This is also allowed. Nothing says that the function has to cover all points in the codomain. Every point in the *domain* must have an arrow coming out of it, but not every point in the *codomain* needs an arrow pointing to it.

Now consider this mapping, which we'll call h:



Is h a function? The answer is no, because one of the points in A does not have an arrow coming out of it. What about this function, which we'll call k:

Remark 14.4. If not every point in the domain has an arrow coming out of it, we call the mapping a **partial function** or **partial mapping**.



Is k a function? The answer is no, because one of the points in A has more than one arrow coming out of it.

14.3 Self-maps

As we have seen, functions map the elements from one set to elements of another set. So far, we have been looking at examples where those two sets are different, e.g., functions from A to B. But there is no reason that a function cannot map elements from one set back to elements in that same set. In other words, there is no reason that we can't have a function from A to A, with the following **signature**:

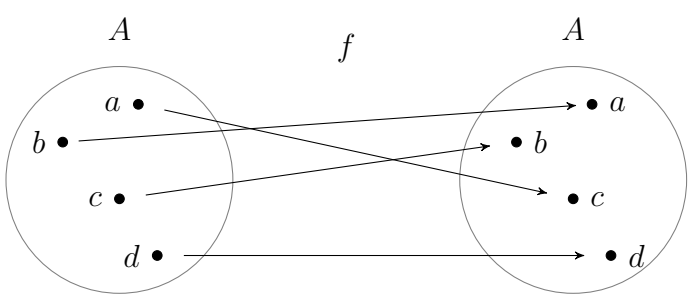
Remark 14.5. There is no reason we cannot have self-map functions. That is, there is nothing to prevent us from constructing functions that map elements in a set back to other elements in that same set.

$$f : A \to A$$

To visualize this, we can draw A twice:

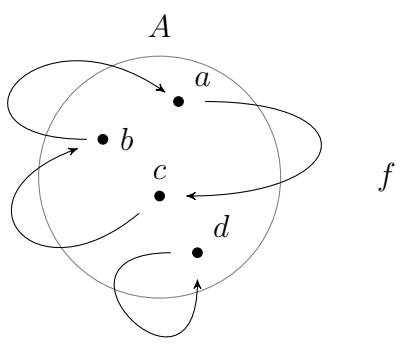
Then, we can draw arrows in. For example, here is one possible function from A to A:

Remark 14.6. If a function maps a set back to itself, e.g., $f : A \to A$, we can draw two copies of the set A, and add the lines. This is one way to depict such a self-mapping function.



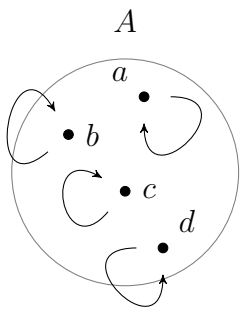
As an alternative way of drawing the same function, we can just draw A one time, and then have the arrows loop back

into the same set, to show which points from A are mapped to which other points in A. Like this:



Remark 14.7. Notice how this drawing depicts the same function $f : A \to A$ that we drew above with two copies of A. That is to say, it maps the same points in A to the same other points in A. This is just a different way of drawing the same function.

There is a special self-map called the **identity function** (or synonymously the **identity map**). This function maps every element in a set back to the very same element. If we draw a picture of it, it looks like this:



This picture makes it clear that the **identity function** for a set maps each element in that set back to itself.

By convention, we call an identity function id, and we add the name of the set it is an identity function for as a subscript. In this case, the set in question is A, so we would call this

Terminology. For any set A, the **identity function** (or as a synonym the **identity map**) on the set A is a function that maps each element in A back to itself. To denote the identity function for A, we write $id_A : A \to A$.

identity function id_A. Hence, the identity function for the set A that we just drew is:

$$id_A : A \to A$$

Similarly, we would denote the identity functions for the sets B and C respectively, as follows:

$$id_B : B \to B \qquad\qquad id_C : C \to C$$

Since an identity function maps each element of a set back to itself, the following is always true:

$$id_A(x) = x$$

Read that expression aloud as "The identity function id_A applied to x gives back x," or "id_A of x is x," but replace "x" with any of the actual elements in A.

Remark 14.8. Since an identity function maps every element in a set to itself, no matter which element in the set we pick, the identity function will give back the same element. Hence, in this example, $id_A(a) = a$, $id_A(b) = b$, $id_A(c) = c$, and $id_A(d) = d$.

14.4 Summary

IN THIS CHAPTER, we learned about the formal definition of a function. We learned that:

- A **function** (also called a **map**) f is a mapping from one set A to another set B. In particular, f maps each element from A to one element from B.
- We denote the **signature** of a function f from the set A to the set B like this: $f : A \to B$. We denote that f maps an element x in A to an element y in B like this: $f(x) = y$. The expression $f(x)$ is a synonym for the element y that f associates with x.

- Functions can also be constructed from a set A to the same set A. For example, $f : A \to A$ is a function f that maps each element of A to another element in A.
- An **identity function** is a function that maps every element in a set to itself. We denote the signature of the identity function for a set A like this: $id_A : A \to A$.

[15]

FUNCTION EQUALITY

IN THE LAST CHAPTER we introduced functions. As we saw, a **function** maps the elements from one set to the elements of another set. To be more exact, a function maps *each* element from the first set to *one* element in the second set.

In this chapter, we will discuss the question of **function equality**. When do we consider two functions to be equal?

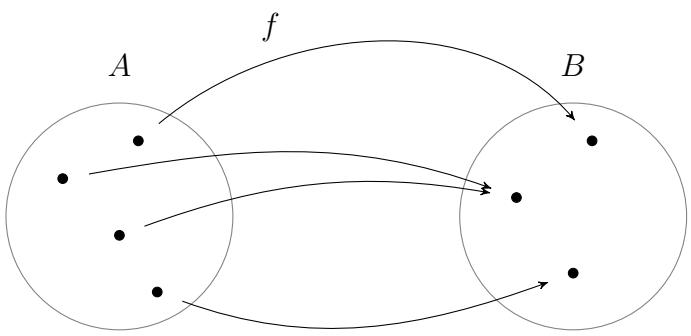
15.1 Equality

WHEN ARE TWO FUNCTIONS THE SAME? The answer is: they are the same if they map the **same keys** to the **same values**. To put it in terms of functions and sets, we consider two functions to be the same if they map the same elements in the

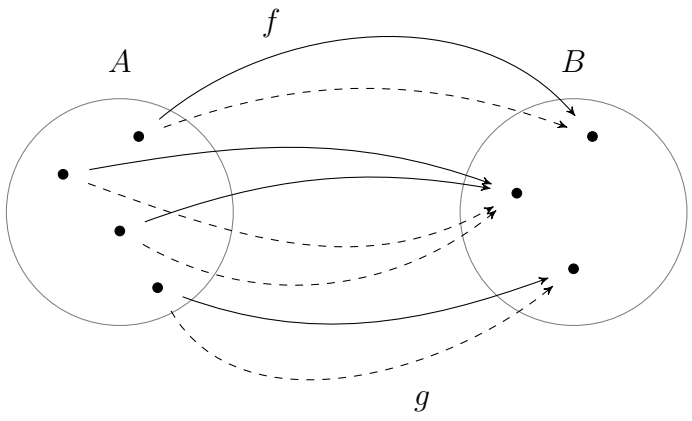
domain to the same elements in the codomain.

For example, suppose we have two sets. For the moment, it doesn't matter which sets we're using, so let's just pick a couple of arbitrary sets, and call them A and B:

Suppose now that we have a function f which goes from A to B. For instance, suppose f looks like this:



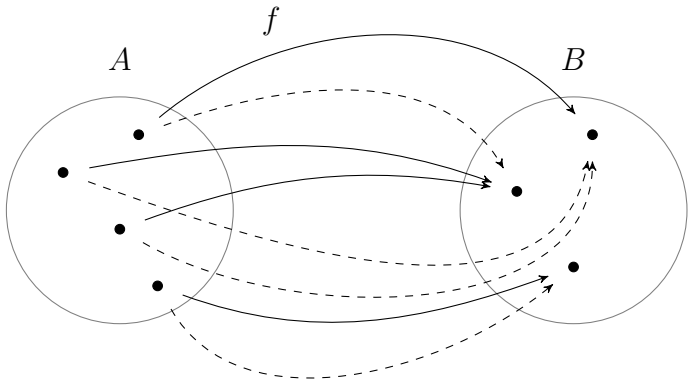
Now suppose that we have another function g, which also goes from A to B. I'll draw g with dashed lines, so that we can tell it apart from f. So, let's suppose that g looks like this:



Now we can ask: are f and g the same? To determine this, we go through each element of A, and we check to see if f and g map each element to the same element in B. In this case, we can follow the arrows, and we can see that both functions do in fact map each element from A to the same element in B. So we can conclude that f and g are the same.

Remark 15.1. To check if two functions are equal, check what they do with each element in the domain. If they are equal, they will map each element in the domain to the same element in the codomain.

Now consider another function h. I'll draw it with dashed lines as well, so that we can tell it apart from f:



Are f and h the same? To determine that, we again need to check each point in A, and see if f and h map it to the same

point in B. From the picture, we can see that f and h do not in fact map all the same points in A to the same points in B. So f and h are not the same.

Let us be a little more formal about how we put this. If we have two functions f and g that map elements from A to elements from B, then f and g are the same if they map every $x \in A$ to the same element in B. That is, if:

$$\text{for every } x \in A, f(x) = g(x)$$

Terminology. Two functions are **the same** or **equal** if they are the same mappings. That is, if they map the same elements in the domain to the same elements in the codomain.

Remember that "$f(x)$" is just a synonym for the element that f maps x to, and "$g(x)$" is just a synonym for the element that g maps x to. So the expression "$f(x) = g(x)$" asserts that whatever f maps x to, it is equal to whatever g maps x to. And then when we put "for every $x \in A$" in front, we are asserting that whatever f maps x to is the same as whatever g maps x, to for *each* element x in the set A.

So let's look at that whole expression again:

$$\text{for every } x \in A, f(x) = g(x)$$

We can read that aloud like this: "for every element x in the set A, the element that f maps x to is the same as the element that g maps x to." Alternatively, you can read it like this: "for every element x in the set A, when f is applied to x and when g is applied to x, they both yield the same element." Or, even more concisely: "for every element x in the set A, f of x is the same as g of x."

When two functions (say, f and g) are indeed the same in this way, then we denote that fact by writing this:

$$f = g$$

Remark 15.2. We use the equals sign to denote that two functions are the same. Hence, you can read "$f = g$" as "f and g are equal as functions," "f is equal to g," or "f is the same as g."

You can read that aloud like so: "f and g are equal functions," or "f and g are the same functions," or even just "f and g are equal."

Let's put all of this down into a definition for function equality.

Definition 15.1 (Function equality)**.** For any two sets A and B and for any functions $f : A \to B$ and $g : A \to B$, we will say that f is equal to g (or synonymously, f is the same as g) when, for every element $x \in A$, $f(x) = g(x)$. If f and g are the same, we will denote that like this: $f = g$.

15.2 Specifying Functions

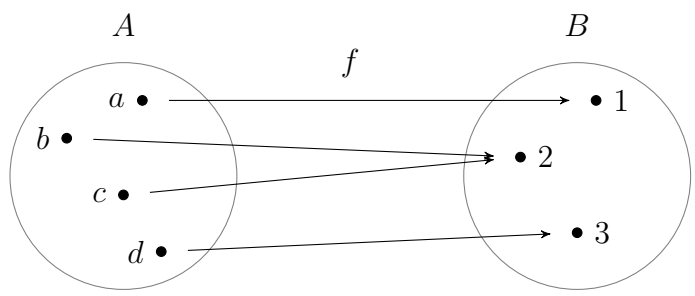
HOW DO WE SPECIFY a function? When we want to talk about a function, how do we specify the function we mean? Here there are no hard and fast rules. The only requirement is that we need to clearly communicate to other people exactly which elements are mapped to which other elements.

We have seen a few ways to do this already. Earlier, we used lookup tables and association lists to present mappings. So, one way to specify a function is to write out its complete lookup table, or to write out the complete association list. So long as its complete, anybody can go and look at your table or list, and see the full mapping.

Another way we have been specifying functions is by drawing pictures of sets, and then drawing arrows from the points in one set to the points in the other. This is also a valid way to specify a function. If we have a picture, then anybody can look at the picture, to see the full mapping.

Another way to specify a function is just to write out every individual mapping explicitly. For example, consider this picture of a function:

Terminology. When we introduced sets, we said that "to specify a set" means to announce to other people which items are in the set we want to talk about. It is similar with functions. If we want to talk to other people about a function, we need to be able to communicate exactly what the internal mappings are that we have in mind. So, to **specify a function** is to announce to others the details of the internal mapping that we have in mind.



We can see that the set A contains four elements, and the set B contains three elements, like this:

$$A = \{a, b, c, d\} \qquad B = \{1, 2, 3\}$$

We can also see that f maps a to 1, it maps b and c to 2, and it maps c to 3. We can just write out all of these mappings, like this:

Notation 15.1. For functions that deal with small sets like A and B here, it is plenty easy to write out all of the internal mappings in this fashion.

let $f : A \to B$ be defined as follows:

$$f(a) = 1 \qquad f(b) = 2$$
$$f(c) = 2 \qquad f(d) = 3$$

That is also a valid way to specify a function. The first line gives us the **signature** of the function, and the rest tells us what f maps each element of A to. This gives everybody all the information there is to know about this function.

Another way to specify a function is to write out its **association list**, as a set. Like this:

$$f : A \to B = \{(a, 1), (b, 2), (c, 2), (d, 3)\}$$

These ways of specifying a function are all plenty easy to do if the domain and codomain are small sets. But what if we are dealing with a really big function, which maps so many elements that there are just too many to list them all out?

In these situations, mathematicians usually just write down a **rule** or **recipe** that tells you how to compute or figure out for yourself which elements the function gives back, for any given input.

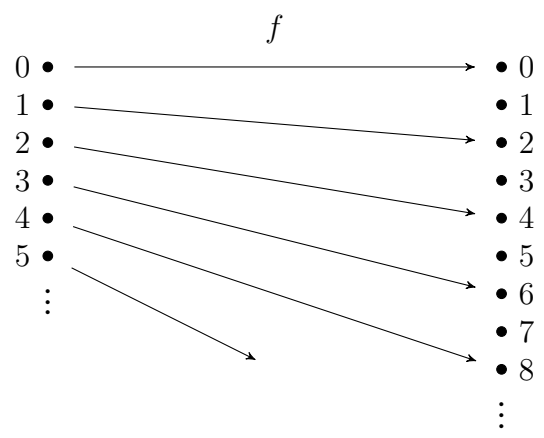
For example, suppose we have a function that looks something like the following. I'll first write it out as a lookup table:

Key	**Value**
0	0
1	2
2	4
3	6
4	8
...	...

Here we have a function, which we can just call f. It has whole numbers as its domain, and it also has whole numbers as its codomain. So its signature is this:

$$f : \text{whole numbers} \to \text{whole numbers}$$

We can see that f maps 0 to 0, 1 to 2, 2 to 4, and so on. The dots in the bottom row indicate that this pattern keeps going, forever. We can draw this function as a picture:



We can specify this function much more concisely by writing down a rule that tells you how to figure out what all the mapped elements are. Notice that each element in the domain is mapped to the number that is twice its size in the codomain. Hence, 1 is mapped to 2, and 2 is mapped to 4, and so on. We can put this down in a nice little rule:

$$\text{let } f \text{ be defined by the rule: } f(x) = 2x$$

What this says is that for any element x in the domain of f, the element that f maps x to can be computed by multiplying x by 2. In other words, for any input x, you can figure out what f maps x to by doubling it.

So, for example, if you want to know what f maps 1 to, double it (1 multiplied by 2 is 2). If you want to know what f maps 2 to, double it (2 multiplied by 2 is 4). If you want to know what f maps 3 to, double it (3 multiplied by 2 is 6).

What does f map 125 to? We can figure it out by using our rule: we just double it. Indeed, 125 multiplied by 2 is 250, so $f(125) = 250$. Hence, even though this function is infinitely large (because it keeps going), we can see that it looks something like this:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 2 \\ f(2) &= 4 \\ f(3) &= 6 \\ f(4) &= 8 \\ &\vdots \\ f(125) &= 250 \\ &\vdots \end{aligned}$$

Here is another example. Let g be defined by the following rule:

$$g(x) = x^2$$

This tells us that for any number x, g is going to map it to the square of that number (i.e., x^2). Hence, we can see that the function looks something like this:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(2) &= 4 \\ f(3) &= 9 \\ f(4) &= 16 \\ &\vdots \\ f(125) &= 15,625 \\ &\vdots \end{aligned}$$

15.3 Equality Again

FUNCTIONS CAN DIFFER in their rules, and still be equal. To see this, consider the following example. Let f and g be defined by the following two rules:

$$f(x) = (x+1)^2 \qquad\qquad g(x) = x^2 + 2x + 1$$

Here f and g are defined by different rules. But are they different functions? How do we check? As we saw above, two functions are the same if they map the same elements in the domain to the same elements in the codomain. So, let's

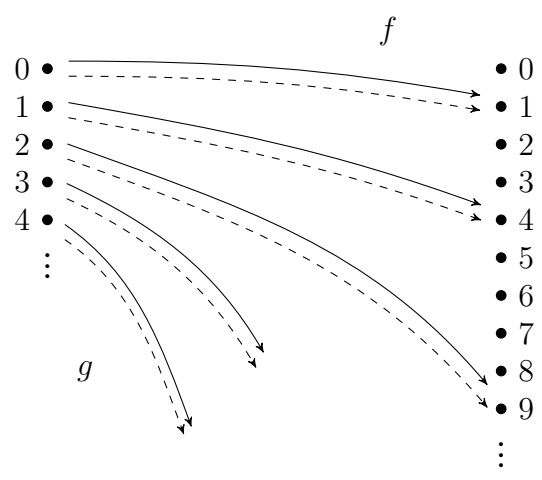
see what these two functions look like, by computing a couple of mappings:

f	
Input	**Output**
0	1
1	4
2	9
...	...
7	64
...	...
24	625
...	...

g	
Input	**Output**
0	1
1	4
2	9
...	...
7	64
...	...
24	625
...	...

Both of these functions appear to map the same inputs to the same outputs, so these two functions look to be the same. We can also draw these two functions, to see it more visually. I'll use solid arrows for f, and dashed arrows for g:

Ponder. To say that f and g are the same functions means that $f(x) = g(x)$ for every x. In other words, it means that f and g map every element from the left side to the same element on the right side.

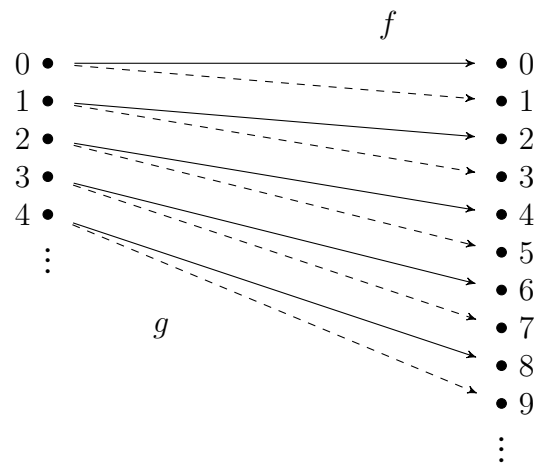


In this picture too, we can see that f and g map each item on the left to the same item on the right. So these two functions look to be equal, even though they have different rules.

Let's consider one more pair of functions. Let f and g be defined by these rules:

$$f(x) = 2x \qquad\qquad g(x) = 2x + 1$$

Are these versions of f and g different functions? As before, we can figure this out by checking that f maps each x to the same element that g does. Let's look at the picture:



We can see from this picture that f and g do not map the elements on the left to the same elements on the right. For instance, f maps 0 to 0 while g maps it to 1, f maps 1 to 2 while g maps it to 3, and so on. So we can conclude that, in this case, f and g are not the same functions.

15.4 Summary

In this chapter, we learned about **function equality**. We learned that two functions are the same (or "equal") when they map the same elements in the domain to the same elements in the codomain.

- More exactly, a function $f : A \to B$ is equal to a function $g : A \to B$ if $f(x) = g(x)$ for every x in A.
- If functions are very large, we can **specify them** by stating a **rule** or **recipe** that tells us how to calculate the mappings ourselves.
- If two functions are specified with different rules, that does not mean they are different functions. If they map the same inputs to the same outputs, they are still the same functions.

[16]

KINDS OF FUNCTIONS

IN THE LAST CHAPTER we learned about **function equality**. We learned that two functions are considered to be equal when they map the same elements from the domain to the same elements in the codomain.

In this chapter, we want to think about what functions look like when the domain and codomain have different sizes (i.e., when they have different **cardinalities**). Certain kinds of functions can be constructed when the domain and codomain are the same size, and other kinds of functions can be constructed when one is bigger than the other.

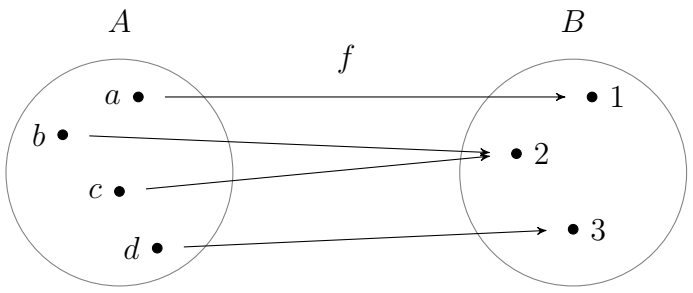
As we will see, there are three special kinds of functions that mathematicians have given special names to: they are **injective functions**, **surjective functions**, and **bijective functions**. Each of these has special characteristics that we will talk about below.

Remark 16.1. Recall that the **cardinality** of a set is the number of elements in it.

16.1 Different Sized Sets

SUPPOSE WE HAVE TWO SETS, A and B, and A is bigger than B. For example, A has four elements, and B has three:

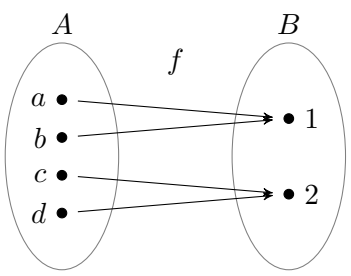
Since there are more elements in A, we can't map each element of A to its own point in B. Some of them are going to have to share. So, for example, we could map both b and c to 2, and then let a and d have their own elements in B, like this:



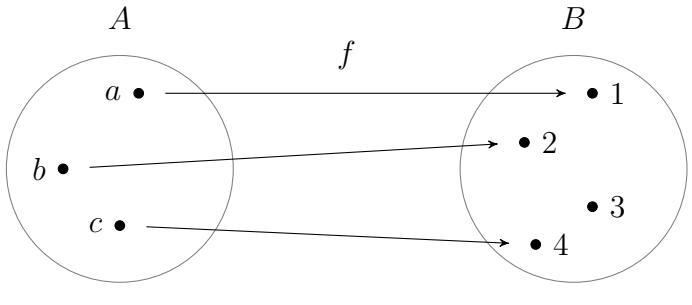
So if the first set has more elements than the second set, then the only way to "fit" all of the points from the first set into the points of the second set is to have some of the arrows **share their endpoints**.

Now consider the opposite scenario. Suppose that B has more elements than A, for instance like this:

Remark 16.2. As A gets bigger and B gets smaller, more and more arrows from A will have to "squeeze in" to fit into B, so to speak. For example:



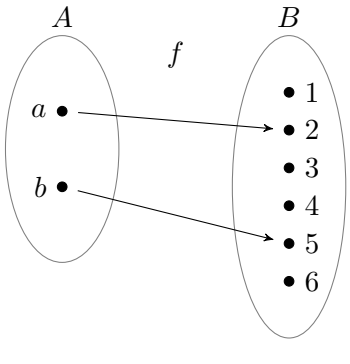
Since there are more elements in B than there are in A, we can't cover all of B with arrows. At least one of the points in B will have to go without an arrow pointing at it. For example, this mapping leaves 3 without an arrow pointing at it:



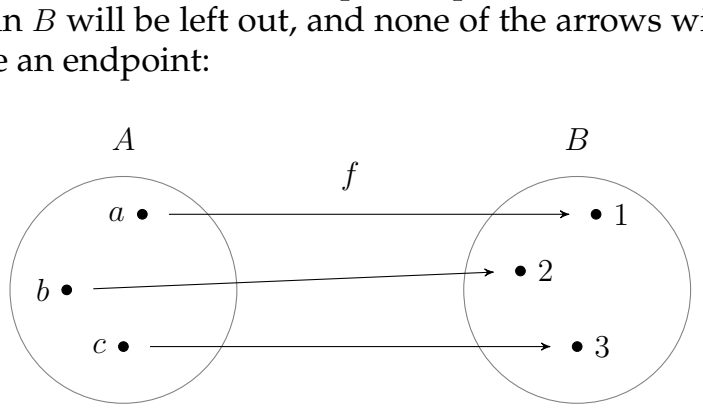
So if the second set is bigger than the first set, then the first set won't be able to cover all of the elements in the second set. At least some points in the second set will **have no arrows pointing to them**.

Now suppose A and B have the same number of elements in them. For instance, suppose both have 3 elements:

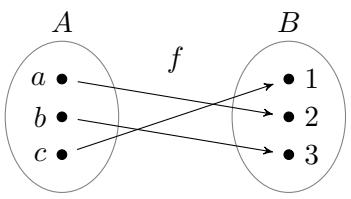
Remark 16.3. As A gets smaller and B gets bigger, there will be more and more points in B that A cannot point to. For example:



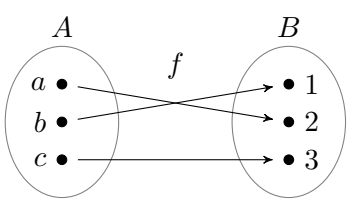
Since both sets have the same number of elements, we can construct a function that maps the points one-to-one. No points in B will be left out, and none of the arrows will have to share an endpoint:



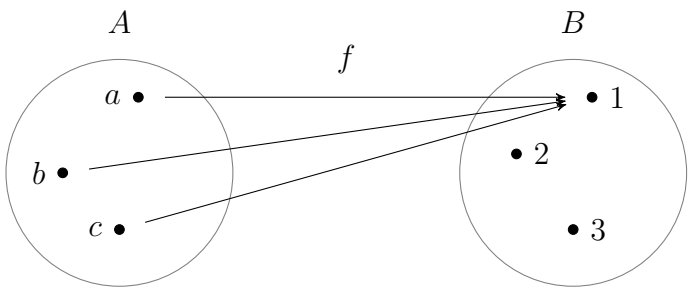
Remark 16.4. There can be multiple one-to-one functions from A to B. Here is one:



And here is another one:



Of course, a function need not cover every point in the codomain. This is a perfectly valid function:

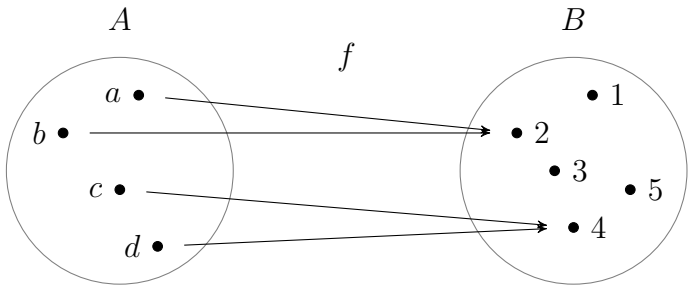


But if both sets have the same size, then it is possible to construct a function that maps the points one-to-one.

16.2 Images

AS WE SAW A MOMENT AGO, a function need not cover all of the points in the codomain. Some functions do cover all of the points in the codomain, but not all functions do.

We can look at any function, and we can make a list of the just the points that have arrows pointing at them. We call this set the **image** of the function. Consider this function:



Terminology. The **image** of a function is the set of elements in the codomain that are mapped by the function. Elements that have no arrow pointing at them are not included in the image.

Which points in B have arrows pointing at them? Just the points labeled 2 and 4. So, let's put these into a set all by themselves. That is the *image* of f:

$$\text{image of } f = \{2, 4\}$$

To denote the image of a function f, we will write this:

$$\mathsf{img}\ f$$

Notation 16.1. To denote the image of a function f, we write "$\mathsf{img}\ f$."

Hence, we can say that the **image** of f is this:

$$\mathsf{img}\ f = \{2, 4\}$$

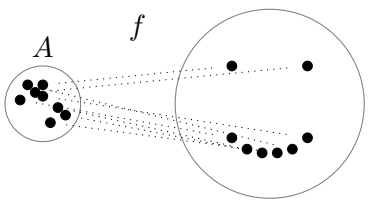
Notice that $\mathsf{img}\ f$ is a **subset** of the codomain B:

$$\mathsf{img}\ f \subseteq B$$

This is because the image makes up a *part* of B. The image is comprised of some, but not all, of the points in B. We might say that f "selects" only certain points out of B.

Remark 16.5. Think of a function as projecting a picture on a projection screen.

Why do we call it the "image" of f? If you like, you can think of the entire codomain as a big projector screen, and you can think of just the points that f points to as the picture or "image" that f "projects" onto the projector screen.



Let's look at one other example. Consider this function:

What is the image of g? In this case, it is 1, 2, and 3. Hence:

$$\mathsf{img}\ g = \{1, 2, 3\}$$

In our previous example, $\mathsf{img}\ f$ was smaller than the codomain. But in this case, $\mathsf{img}\ g$ makes up the entire codomain of g. But that's okay. The image is still a subset of the codomain, because a subset can be equal to its superset. Hence, this is true:

$$\mathsf{img}\ g \subseteq B$$

Remark 16.6. Recall that if A is a subset of B, i.e., $A \subseteq B$, then A can either be smaller than B, or it can be equal to B. A subset is analogous to being "less than or equal to."

And, in this case, this is true too:

$$\mathsf{img}\ g = B$$

Let's put down a proper definition for images. To do that, let's get a little bit more formal.

We can find the image of a function f by taking $f(x)$ for every x in the domain. Recall that "$f(x)$" is just a synonym for the element in the codomain that f maps x to. So, if we go through every x in the domain, and find $f(x)$ for that x, then we will have all of the elements that make up the image. With that, we can construct a definition:

Definition 16.1 (Images)**.** For any function $f : A \to B$, we will say that the **image** of f is the set that is comprised of every element $f(x) \in B$, for every $x \in A$. To denote the image of f, we will write this: $\mathsf{img}\ f$.

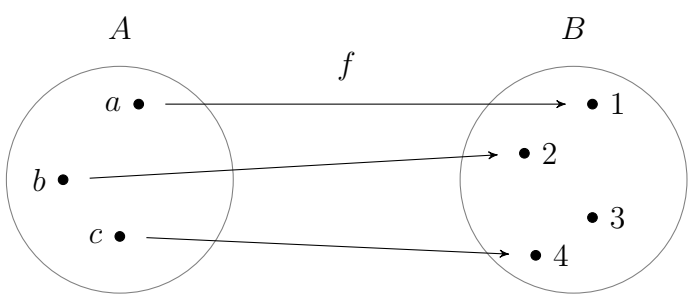
16.3 Injective Functions

IF A FUNCTION MAPS the points of its domain to distinct endpoints, then we say it is *injective*. An **injective function** is a function whose arrows all point to distinct endpoints. None of its arrows point to the same endpoint.

As a slogan, we might say that an injective function **preserves distinctness**. It preserves the distinctness of its original points, by not mapping any two points to the same endpoint.

Another slogan we might use is this: an injective function **never collapses paths**, i.e., if the arrows are "paths" from the one set to the other, then an injective function never has two paths that go to the same point.

As an example, consider this function:

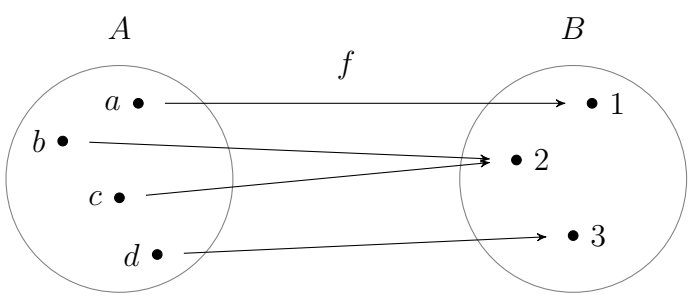


Terminology. An **injective function** maps each point in the domain to a distinct endpoint in the codomain. No two arrows point to the same endpoint in the codomain. It preserves the distinctness of points, and never collapses paths.

This function is injective, because each point from the do-

main is mapped to a distinct point in the codomain. None of the arrows go to the same place in the codomain.

Here is an example of a function that is *not* injective:



This is not injective, because two arrows point to the same endpoint (b and c are both mapped to 2). So this function collapses a path (it collapses b and c to 2), and hence it does *not* preserve the distinctness of its original points.

Let us write down a definition for injective functions. We can say that, if a function is injective, then it will satisfy this condition: if $f(x)$ and $f(y)$ in the codomain are the same, then x and y in the domain must be the same.

Terminology. Some mathematicians call an injective function a **one-to-one**-function, because it maps each item in the domain to a separate item in the codomain. You never get a two-to-one mapping, where it maps two elements in the domain to the same element in the codomain.

Ponder. No function from a bigger set to a smaller set can be injective. Can you see why?

Remark 16.7. An equivalent way to define injective functions would be to put it this way: if $f(x) \neq f(y)$, then $x \neq y$.

Definition 16.2 (Injective functions)**.** For any sets A, B and any function $f : A \to B$, we will say that f is **injective** if, for any $x, y \in A$, if $f(x) = f(y)$, then $x = y$.

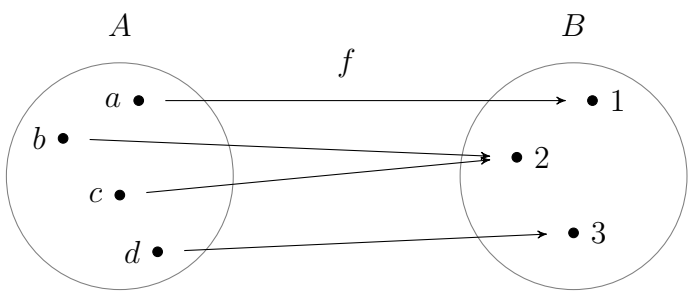
16.4 Surjective Functions

IF A FUNCTION COVERS all of the codomain with arrows, then we say it is *surjective*. A **surjective** function is a function that sends an arrow to each and every point in the codomain.

It leaves no points in the codomain without an arrow pointing at them.

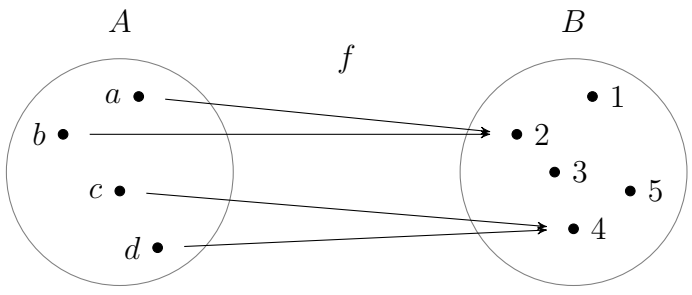
As a slogan, we can say that a surjective function **completely covers** the codomain. When it maps the points of the domain to the codomain, it doesn't miss any points in the codomain. It gets each one.

Here is an example of a surjective function:



This function is surjective because every point from the codomain has an arrow pointing at it.

Here is an example of a function that is *not* surjective:



This function is not surjective, because it does not cover the entire domain. It leaves the points 1, 3, and 5 with no arrows pointing at them.

Notice that with a surjective function, the **image** of the function is equal to its codomain. Since a surjective function is a function that sends an arrow to every point in the

Terminology. A **surjective** function covers the entire codomain: it sends an arrow to every point in the codomain. No codomain points are left without an arrow pointing at them.

Remark 16.8. Note that this function is *surjective*, but it is not *injective*.

Remark 16.9. No function from a smaller set to a bigger set can be surjective. Can you see why?

Terminology. Some mathematicians call a surjective function an **onto**-function, and they say that it maps its domain *onto* its codomain. So, an injective function is sometimes called a *on-to-one*-function, whereas a surjective function is sometimes called an *onto*-function.

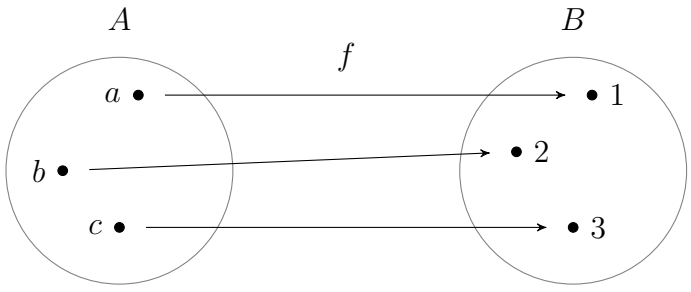
codomain, the image of a surjective function is just the entire codomain. We can use that to define surjective functions:

Definition 16.3 (Surjective functions)**.** For any sets A, B and any function $f : A \to B$, we will say that f is **surjective** if img $f = B$.

16.5 Bijective Functions

A FUNCTION CAN BE BOTH injective and surjective. If a function is both, we say it is **bijective**. A bijective function essentially pairs up the elements of the domain and codomain in a one-to-one way. Here is an example of a bijective function:

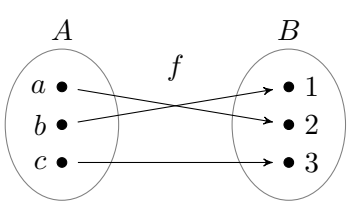
Terminology. A **bijective** function is a function that is both injective and surjective. In other words, the elements of the domain and codomain stand in a one-to-one relationship. Because it is both injective and surjective, it is sometimes called a **one-to-one-and-onto**-function.



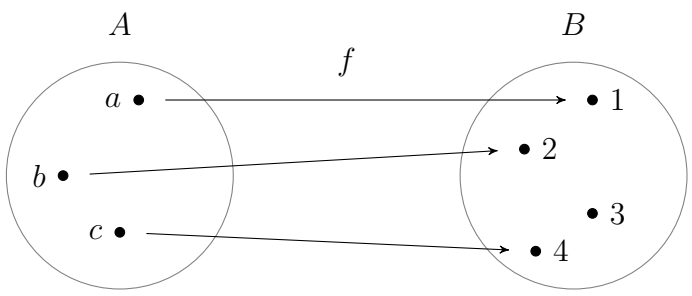
This function is bijective because we can see that each point in the domain and codomain are paired up, one-to-one.

Why is a bijective function a one-to-one mapping? It is because it is *both* surjective and injective. Since it is surjective, there has to be an arrow going to *each* point in the codomain. But since it is injective, no two arrows can go to the *same* point in the codomain. So every point in the codomain will be covered, but every point will have its own arrow going to it. Hence, the mapping has to be one-to-one.

Remark 16.10. There is more than one way to construct a bijective function. Here is another one:



Here is an example of a function that is *not* bijective:



Remark 16.11. Bijective functions can only be constructed between sets that have the same cardinality (i.e., the same number of elements). Can you see why?

This function is not bijective, because the elements are paired up in a one-to-one way. Three of the elements *are* paired up one-to-one, but the element 3 is a loner. It has nothing paired up with it. A function is bijective only if *all* of the elements are paired up in a one-to-one fashion.

Let's put this down in a definition.

Definition 16.4 (Bijective functions)**.** For any sets A, B and any function $f : A \to B$, we will say that f is **bijective** if it is both injective and surjective.

16.6 Summary

IN THIS CHAPTER, we learned about three special kinds of functions.

- A function is **injective** if it preserves distinctness, and does not collapse any paths. That is to say, each element in the domain gets mapped to a distinct element in the codomain. No two arrows end up at the same endpoint.

- A function is **surjective** if it covers the entire codomain. That is to say, each element in the codomain has an arrow pointing at it.
- A function is **bijective** if it is both injective and surjective. A bijective function is a one-to-one mapping: it puts the elements from the domain and the codomain in a one-to-one correspondence.

[17]

ISOMORPHISM

IN THE LAST CHAPTER we learned ...

17.1 Inverses

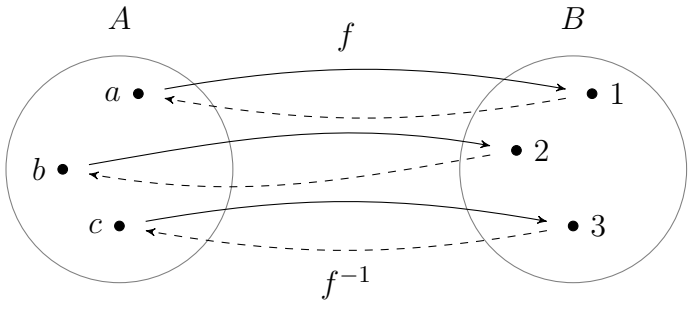
AN INTERESTING FACT about bijective functions is that they are reversible. For any bijective function f that goes from A to B, we can construct another function from B to A that takes us right back to the same points we started with. We call such a function the **inverse** of f, and we denote it like this:

$$f^{-1}$$

Let's look at a picture of what this looks like, so we can see it in more detail. The following is a picture of a function

Terminology. A function f is **reversible** if we can construct another function that takes us back to f's starting points. We denote the inverse of f like this: f^{-1}.

f and its inverse f^{-1}. I've drawn f with solid lines, and f^{-1} with dashed lines, so we can tell them apart:



The defining feature here is that we can use these two mappings to go back and forth, and we always end up right where we started. Pick any x from A and follow the arrow to $f(x)$. Then, use f^{-1} to go back. You should end up right back at x. We can state that like so:

Remark 17.1. A bijective function and its inverse are characterized by the fact that we can use them to go from one set to the other, then back again, and we will always end up right where we started.

$$\text{for any } x \in A, \text{ if } f(x) = y, \text{ then } f^{-1}(y) = x.$$

Read that aloud like so: "for any element x in the set A, if the function f maps x to y, then the inverse function f^{-1} maps y back to x." Or, to be more concise, you could say: "for any x in A, if f maps x to y, then f^{-1} maps y to x."

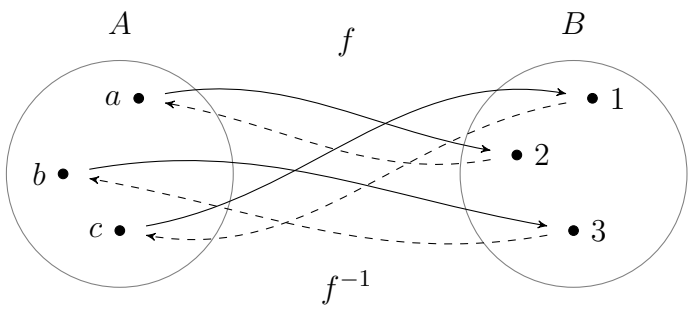
It's the same the other way too, if we start from B. Pick any y from B, and follow the arrow to $f^{-1}(y)$. Then, use f to go back. You should end up right back at y. We can state this point too, like so:

$$\text{for any } y \in B, \text{ if } f^{-1}(y) = x, \text{ then } f(x) = y.$$

Let's put all of this down in a definition of inverses.

Definition 17.1 (Inverses)**.** For any sets A, B and function $f : A \to B$, we will say that there is an **inverse** of f, which we will denote as f^{-1}, if the following two conditions hold: (i) for any $x \in A$, if $f(x) = y$, then $f^{-1}(y) = x$, and (ii) for any $y \in B$, if $f^{-1}(y) = x$, then $f(x) = y$. If f has an inverse f^{-1}, we will say that f is **reversible**.

Here is another bijective function with its inverse:



This picture looks a little more complicated than the previous one, but this is only because the arrows are crossing each other. But nevertheless, this is a bijection and its inverse.

We can check that this is so by following the arrows from one side and then back, and confirming that we always end up back at the same starting point. And indeed, if you do this check yourself, you'll find that no matter which side you start on, you'll always end up back at the same point. For instance, $f(a)$ goes to 2, and $f^{-1}(2)$ goes right back to a. Similarly, $f^{-1}(3)$ goes to b, and $f(b)$ goes right back to 3.

Remark 17.2. To check whether a bijective function and a function going the other way truly are inverses, check **both ways**. That is, check that if you follow the arrows out and then back, you end up right back where you started, for each point on the left side, and then for each point on the right side.

17.2 Isomorphism

IF TWO SETS have a bijection (and hence also have an inverse), then that lets us go back and forth between them. When this is so, we say that the two sets are **isomorphic** to each other.

Terminology. If two sets have a bijection and inverse between them, we say they are **isomorphic**, which means they have the same shape.

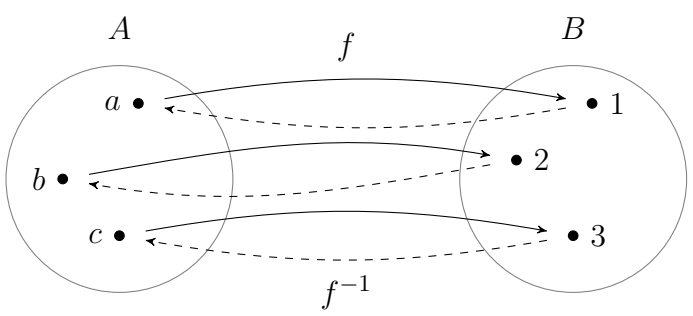
The word "isomorphic" derives from the Greek words "iso" and "morphe," meaning "same" and "shape." In essence, if there is a bijection between two sets, then this tells us that those two sets have the **same shape**.

Let's think about that for a moment. If we have a bijection (and hence also its inverse), then this tells us that, in fact, the two sets do have exactly the same shape. After all, there is a one-to-one correspondence between them. For each element in the one, there is a corresponding "twin" in the other.

Remark 17.3. If two sets have a bijection, then this means that each element in the one has a "twin" in the other. So the two sets are basically just mirror images of each other, as it were.

Of course, the *names* of the elements inside the two sets may be different, but given that every element in the one has a twin in the other, it follows that the basic shape of these two sets are completely the same.

As an example, consider these isomorphic sets:



Remark 17.4. If two sets have the same shape, then the only real difference between them is just the **names** of the elements.

These two sets have different names for their elements. In A the elements are named a, b, and c, whereas in B they are named 1, 2, and 3. But those are just names.

But the fact that we have a bijection and its inverse shows us that we can "translate" from one to the other seamlessly. The bijection is basically just a name translator. We might want to call the element "a," but we can translate it to "1" (and we can translate "1" back to "a").

So to say that these two sets are isomorphic, is to say that they have exactly the same shape and they just have different names (which we can translate with the bijection and its inverse).

Because a bijection tells us that the two sets are isomorphic, mathematicians sometimes call a bijection an *isomorphism*. An **isomorphism** is just a bijective function that has an inverse.

Let us put all this down as a definition.

Terminology. A bijection (or its inverse) is called an **isomorphism** because it shows us that the two sets in question are isomorphic. Hence, the terms "bijection" and "isomorphism" are synonyms.

Definition 17.2 (Isomorphisms)**.** For any sets A, B and any function $f : A \to B$, we will say that f is an **isomorphism** if f is a reversible bijection. If there is an isomporphism f between A and B, then we will say that A and B are **isomorphic**, and we will denote that like this: $A \cong B$.

17.3 Sameness up to Isomorphism

Isomorphism gives us another way to talk about how two sets are "the same." Earlier, we learned about **set equality**: two sets are *equal* if they have exactly the same elements. But often, we don't really care what the elements are called. Really, if the two sets have the same shape, that's good enough.

To look at our last example, suppose we remove the names, and just look at the two sets as bags of dots. The following

Remark 17.5. Set equality is a very strict kind of equality: it requires that the two sets in question have *exactly* the same elements (and that even means that the elements must have the same names).
Isomorphism lets us talk about a looser kind of sameness.

picture is what that looks like, and we can see that our two sets really do have the same shape:

To capture the idea that these two sets are "the same" in this looser sense (i.e., in the sense that they have the same shape, and they differ only in the names of their elements), we say that they are the same **up to isomorphism**.

Here we have hit upon a concept in math that occurs over and over again. In math, we are interested in structures that have the *same shape*. That is, we are interested in structures that are *isomorphic*.

How do we determine if two structures have the same shape? We can tell that they are isomorphic if we can construct an isomorphism — that is, if we can construct a function that maps each part of one to a "twin" part of the other, and then we can construct another function that maps them back again to exactly the same starting points. If we can do that, then we know the two structures are isomorphic.

Mathematicians often call such isomorphisms that "translate" from one structure to another and back again **structure preserving maps** or **structure preserving functions**, because they are maps that preserve the basic structure of the things they are translating between. We will encounter structure-preserving maps repeatedly as we proceed.

Remark 17.6. This is a key idea in math that we will see over and over again. In many areas of math, we want to establish (a) what the **structures** that we are studying look like, and (b) what the **structure preserving maps** look like for these structures.

17.4 Summary

IN THIS CHAPTER, we learned about a new way to think about "sameness" of sets.

- If there is a bijection f that goes from A to B, then there is an **inverse** function that goes from B to A and takes us right back to where we started. The inverse function of f is denoted f^{-1}.
- If two sets have a bijection (and hence an inverse) between them, then we say they are **isomorphic**. The fact that there is a bijection between them tells us that they have the very same shape, and differ only in the names of their elements.
- Because a bijection translates between isomorphic sets, we can also call a bijection an **isomorphism**.
- The idea of **isomorphic structures** and **structure preserving maps** that we can use to translate between them is a far-reaching idea that occurs everywhere in math.

[18]

FURTHER READING

To pursue functions further, the following list may offer some helpful starting points.

- Stewart and Tall (2015, ch. 5) provides a good but introductory-level discussion to functions.
- Steinhart (2018, ch. 2) also provides an excellent introduction for the beginner to the topic of functions (and relations).
- Stewart (1995, ch. 5) offers a discussion of functions for non-experts.
- Flegg (1974, ch. 15) provides a conceptual discussion of functions.
- Cummings (2020, ch. 8) presents a slow, thorough discussion of functions and how to construct proofs about

them.

- Zach (2019, ch. 3) offers a nice, introductory (yet still technical) discussion of functions.
- Velleman (2019, ch. 5) provides an excellent introduction to the concept of functions and proofs about them.
- Madden and Aubrey (2017, chs. 11–12) give thorough coverage of functions and proofs about them.
- Warner (2019, ch. 4) provides thorough coverage of functions and proofs about them.
- Pinter (2014, ch. 2) offers a simple discussion of many basic theorems and proofs about functions.
- Jongsma (2019, ch. 6) provides a rigorous introduction to some of the basic theorems and proofs about functions (and equivalence classes).

Part [4]

RELATIONS

Functions are special kinds of mappings, because they map each object in a domain to one object in a codomain. The key feature of a function is that it pairs up *every* object in the domain with an object in the codomain. **Relations** are looser sets of pairings. A relation need not pair up *every* object in the domain. It can skip some. Or, it can pair an object up with more than one other objects. A relation is basically just any pairing of objects from sets, no matter how loose or dense the pairings happen to be.

In Part 4, we discuss **relations**. We discuss what they are, how they are defined, and many of their properties. We also see how they (along with sets and functions) help us define mathematical **structures**.

[19]

RELATIONS

A FUNCTION IS A SPECIAL KIND OF MAPPING. A mapping qualifies as a function only if it satisfies some fairly strict conditions: in particular, it must map **each** element in the domain to **one** element in the codomain. However, sometimes we might want to relax these restrictions, and allow looser mappings.

For instance, we might want to remove the restriction which says that we must specify a mapping for *each* item from the domain. Instead, we might want to give ourselves permission to skip some of the items in the domain, if we so please, and not map them to anything. In other words, we might want to allow some points in the domain to have no arrows coming out of them at all.

Similarly, we might want to remove the restriction which says that we must map each item from the domain to *only one*

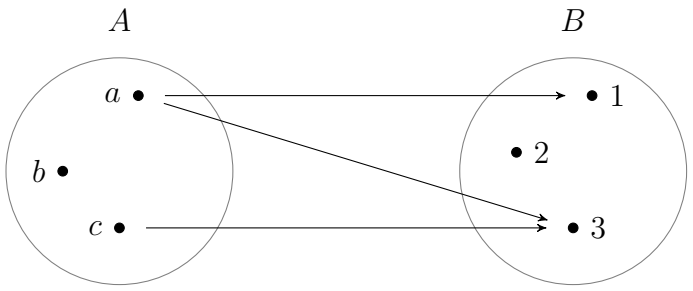
item in the codomain. Instead, we might want to give ourselves permission to map a point from the domain to *many* points in the codomain. That is to say, we might want to allow multiple arrows to come out of a point in the domain, if we so please.

This more relaxed notion of a mapping is called a **relation**. A relation is really just *any* mapping from some points in the domain to some points in the codomain. In this chapter, we will discuss relations, and their properties.

19.1 Looser Mappings

AS WE SAID A MOMENT AGO, a relation is a much looser sort of mapping than a function. In essence, a **relation** is any pairing of elements from the domain with elements from the codomain. Consider this mapping:

Terminology. A **relation** is any mapping of elements from the domain to the codomain. It does not need to satisfy the requirements of a function, that is to say, it does not need to provide exactly *one* mapping for *each* element in the domain. A relation can pair up any elements it likes, as many times as it likes, so to speak.



Notice that a is mapped to 1 and also to 3, while c is mapped to 3. But that's it. There are no further arrows here. This mapping fails to qualify as a function, in two separate ways:

- It is required of a function that it provide a mapping for *each* element in the domain. But here, there is no mapping for b, so this cannot qualify as a function.

- It is also required of a function that it provide *only one* mapping for each element in the domain. But here, there are *two* mappings for a: a is mapped to 1, but also to 3. So this cannot qualify as a function for this reason too.

So this is a *relation*, and not a function. Nevertheless, this is still a mapping in the sense that it pairs up items from A with items in B.

Let's put this into **lookup table**:

A	B
a	1
a	3
c	3

Like the lookup tables we discussed earlier, this one pairs up "keys" (in the left column) with "data" (the right column). However, we can see that this is a table for a relation rather than a function, because:

- Each "key" is not unique (there are two rows for "a").
- Not every item in the domain has a row in this table (there is no row for b).

Now let's present this same mapping as an association list:

$$\begin{aligned}\mathsf{relation} = &(a, 1),\\ &(a, 3),\\ &(c, 3).\end{aligned}$$

Here too we can see the same information. We can see that this mapping pairs up a with 1, and also with 3, and it pairs up c with 3.

Remark 19.1. A mapping can fail to be a function in two ways: if it doesn't provide a mapping for an element in the domain, or if it maps an element from the domain to more than one item in the codomain.

Remark 19.2. Recall that an **association list** is just a list of the pairings, with the first item in each pair being the "key" and the second item in each pair being the "data."

19.2 Definition and Notation

We can give names to relations, just as we could give names to functions. We noted before that mathematicians tend to name their functions f, g, and h. They tend to name relations R, S, and T.

Notation 19.1. We can give relations **names**, and we tend to name them R, S, and T. To denote that a relation R **pairs up** a particular x to y, we write this: $R(x, y)$.

With functions, when we want to denote that a function f maps a particular element x to an element y, we write this:

$$f(x) = y$$

With relations, we write it a little differently. To denote that a relation R maps a particular element x to another element y, we write this:

$$R(x, y)$$

Notation 19.2. Sometimes, mathematicians don't write $R(x, y)$. Instead they write xRy. For example, instead of writing $R(c, 3)$, they would write $cR3$.

Read that aloud like so: "The relation R relates x to y," or "x is related to y by the relation R." Look back to the picture we drew above. Let's give that relation the name "R." We can see from the picture that R relates c to 3. So we could write that like this:

$$R(c, 3)$$

And we read that like this: "R relates c to 3," or "c is related to 3 by R." In the same way, we can also use this notation to assert that "R relates a to 1," and "R relates a to 3." Like this:

$$R(a, 1) \qquad\qquad R(a, 3)$$

Remark 19.3. As with functions, there are no hard and fast rules about how to specify a relation. What's important is just to be absolutely clear about which elements the relation pairs together, so that other people know exactly what the mapping is that you have in mind.

We can depict a relation in many ways. For instance, we can depict it as a **table** or an **association list**, like we did a moment ago. We can also specify a **rule** or **recipe**, that tells other people how to calculate the pairs themselves, just as

we can do with functions. Or we can depict a relation with a picture, like we did earlier.

But if we look underneath all these particular modes of presentation, we see that a relation between two sets A and B is really just a set of pairings of elements from A and B.

Hence, we can write out a relation as a set of pairs. For example, the relation R pictured above can be written out like this:

$$R = \{(a, 1), (a, 3), (c, 3)\}$$

Notice that this is a **subset** of the **product** $A \times B$. What is the product of A and B? It is all possible pairings of elements from A and B. Here is the product (presented as a grid, so it's easy to see all the pairings):

	1	2	3
a	$(a, 1)$	$(a, 2)$	$(a, 3)$
b	$(b, 1)$	$(b, 2)$	$(b, 3)$
c	$(c, 1)$	$(c, 2)$	$(c, 3)$

Compare our relation to this grid of pairings. You can see that our relation $\{(a, 1), (a, 3), (c, 3)\}$ is just a subset of the coordinates we have in this grid. We've just picked out three of the pairings from this grid. Here they are, highlighted:

	1	2	3
a	$\mathbf{(a, 1)}$	$(a, 2)$	$\mathbf{(a, 3)}$
b	$(b, 1)$	$(b, 2)$	$(b, 3)$
c	$(c, 1)$	$(c, 2)$	$\mathbf{(c, 3)}$

This shows that when we construct a relation, we are really just picking out, or selecting, some of the pairs from the product. So a relation between a set A and another set B is really just a subset of the product $A \times B$. To put this into symbols:

Remark 19.4. Recall that the **product** of A and B is the set of all possible pairings of points from A and B.

Remark 19.5. A **function** is also a **subset of the product**, but a function must satisfy the additional requirement that **each** element from the domain must be paired up with **one** element from the codomain. Relations do not have this restriction. A relation is just *any* subset of the product.

$$R \subseteq A \times B$$

Read that aloud like this: "R is a subset of the product of the set A and the set B."

When we discussed functions, we noted that we can describe the **signature** of a function like this: $f : A \to B$. That tells us (i) the name of the function, (ii) the domain, and (iii) the codomain.

Relations also have a **name**, a **domain**, and a **codomain**, so we can describe relations with a signature too. To denote the **signature** of a relation, let's use the subset notation we used a moment ago. Hence, to describe the signature of the relation R from A to B, we will just write this:

$$R \subseteq A \times B$$

Let's put down everything we have said here into a definition for relations.

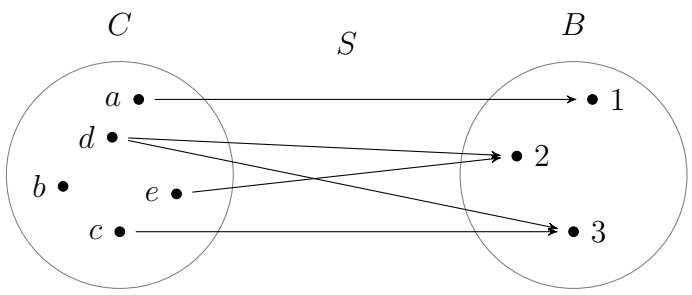
Remark 19.6. Recall that the **signature** of a function $f : A \to B$ tells us that the function is called "f," it tells us that the domain is the set A, and it tells us that the codomain is the set B.

Notation 19.3. To denote the **signature** of a relation R from a set A to a set B, we write this: $R \subseteq A \times B$.

Definition 19.1 (Relations)**.** For any sets A and B, we will say that a **relation** from A to B is any set of pairs (x, y), where x is an element taken from A and y is an element taken from B. To denote the signature of a function R from A to B, we will write this: $R \subseteq A \times B$. To denote that R pairs up a particular element x from A with a particular element y from B, we will write this: $R(x, y)$.

19.3 Examples

Example 19.1. Here is an example of a relation $S \subseteq C \times B$:



We can also write this out as a set of pairs:

$$S = \{(a, 1), (c, 3), (d, 2), (d, 3), (e, 2)\}$$

We can say that the relation S relates c to 3, or d to 3, or e to 2, like this:

$$S(c, 3) \qquad S(d, 3) \qquad S(e, 2)$$

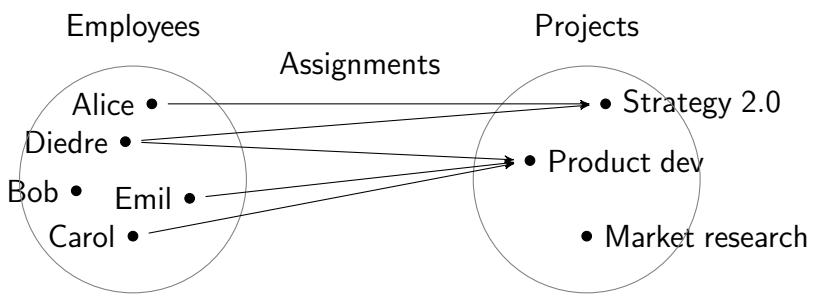
Example 19.2. Suppose we have some employees:

$$\mathsf{Employee} = \{\mathsf{Alice}, \mathsf{Bob}, \mathsf{Carol}, \mathsf{Diedre}, \mathsf{Emil}\}$$

Suppose we also have some projects that the company is working on:

$$\mathsf{Project} = \{\mathsf{Strategy\ 2.0}, \mathsf{Product\ dev}, \mathsf{Market\ research}\}$$

We can assign employees to projects:



Remark 19.7. Notice that the set labeled as Employees is isomorphic to the one pictured as C above. This one just has different names: instead of a it's $Alice$, instead of b it's Bob, and so on. Likewise, the set Projects is isomorphic to the one pictured as B above. It just has different names too.

This is just a relation, of course, from Employees to Projects. Hence it has the signature:

$$\mathsf{Assignments} \subseteq \mathsf{Employees} \times \mathsf{Projects}$$

In the picture, we can see that Alice and Diedre are assigned to "Strategy 2.0," while Diedre, Emil, and Carol are assigned to "Product dev." We can also see that Bob has no projects to work on, and nobody is working on "Market research."

We can write this all out as a set of pairs too:

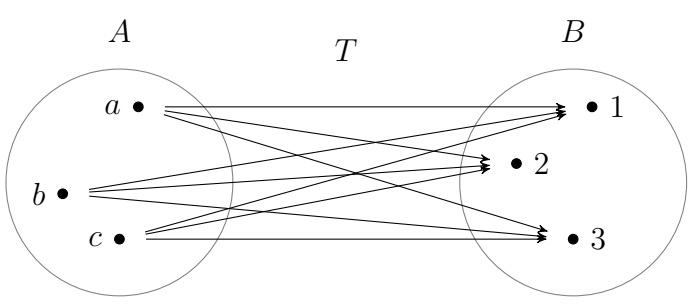
$$\begin{aligned}\mathsf{Assignments} = \{&(\mathsf{Alice}, \mathsf{Strategy\ 2.0}), (\mathsf{Diedre}, \mathsf{Strategy\ 2.0}),\\ &(\mathsf{Diedre}, \mathsf{Product\ dev}), (\mathsf{Carol}, \mathsf{Product\ dev}),\\ &(\mathsf{Emil}, \mathsf{Product\ dev})\}\end{aligned}$$

We can use our special notation to say things like "Alice is assigned to Product dev" (or to put it another way: "the Assignments relation relates Alice to Product dev"):

$$\mathsf{Assignments}(\mathsf{Alice}, \mathsf{Product\ dev})$$

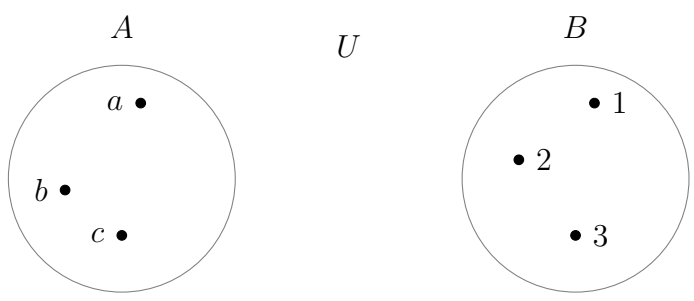
Remark 19.8. Is Assignments a *function*? No, it is not. Why not? There are two ways it fails to be a function. First, there is an item in the domain that has no arrow coming out of it: Bob is not assigned to any project. Second, some people in the domain have more than one arrow coming out of them: Diedre, for example, is assigned to two projects.

Example 19.3. An extreme example of a relation is one where *all of the items* in the domain and codomain are related to each other. For instance:



Here, each item in the domain A is related to all of the items in the codomain B. There are three items in the codomain, and so each item in A has three arrows coming out of it. So this relation has every possible pairing between A and B. (Notice that this is identical to the **product** of A and B.)

Example 19.4. Another extreme example of a relation is one that relates *none of the items* in the domain and codomain. For example, consider this one:



Remark 19.9. An empty relation may seem like a useless relation, but it's always important to think of the **limit cases**, just to make sure that we understand the concept fully. In this context, the limit cases are (i) the case where there are **all** possible arrows, and (ii) the case where there are **no** arrows.

This relation has no arrows going from A to B. Hence, the list of pairings in this relation is entirely empty:

$$U \subseteq A \times B = \{\} \qquad \text{i.e.} \qquad U \subseteq A \times B = \varnothing$$

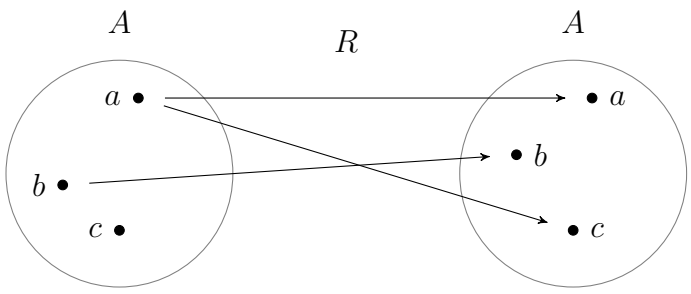
Read that aloud like so: "the relation U from A to B is empty," or "the subset U of the product $A \times B$ is empty."

19.4 Self-relations

WHEN WE DISCUSSED FUNCTIONS, we pointed out that there can be self-maps. That is to say, there is nothing in the definition of a function that stops us from constructing a function

that maps a set back to itself. The same goes for relations. We can have relations from a set to itself.

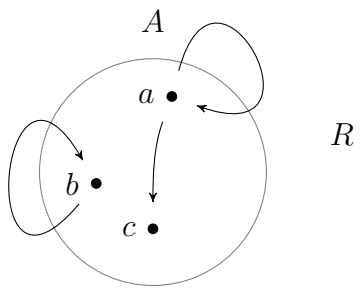
To draw such a relation, we can do it in two ways. We can draw another copy of the set in question, and then draw our arrows in. For instance, here is a relation $R \subseteq A \times A$:



Terminology. When we talk about **self-relations**, i.e., relations that relate a set A to itself, we will sometimes say that it is "a relation **defined over** A," or "a relation **defined over** A," or even more simply that it is "a relation **on** A" or "a relation **over** A." All of these are equivalent ways of saying it relates the set A *to itself*.

This is a relation from A to A, so it is a relation that relates some of the elements in A to other of the elements in A.

Another way to draw the same relation is to just draw one circle, and then have the arrows go out and come right back. Like this:



This picture represents the very same relation, just in a different way. Whichever picture we choose, we can write out the pairings directly:

$$R = \{(a, a), (a, c), (b, b)\}$$

19.5 Summary

IN THIS CHAPTER, we learned about **relations**, which are looser mappings than functions.

- A **function** is a mapping from one set to another, but it is subject to the constraint that it must map *each* element in the domain to *one* element in the codomain. A **relation** is a mapping too, but without that constraint. A relation need not pair up each item in the domain, nor must it pair it with only one item in the codomain.
- Formally speaking, a relation R from A to B is any set of pairs (x, y), where x comes from A and y comes from B. Hence, a relation from A to B is just a **subset of the product** $A \times B$.
- To describe the **signature** of a relation R from A to B, we write this: $R \subseteq A \times B$. To denote that R relates x to y, we write this: $R(x, y)$.
- A relation can relate items from a set to itself, just as functions can. Hence, there can be a relation R from the set A back to the set A, i.e., $R \subseteq A \times A$.

[20]

PROPERTIES OF RELATIONS

In Chapter 19, we looked at **relations**. A relation R from a set A to a set B is any pairing of elements from A and B (where the first item in each pairing comes from A and the second comes from B). We can construct relations from one set A to a another set B, but we can also construct relations from a set A to itself. We might call this latter type of relation a **self**-relation, because it relates a set to itself.

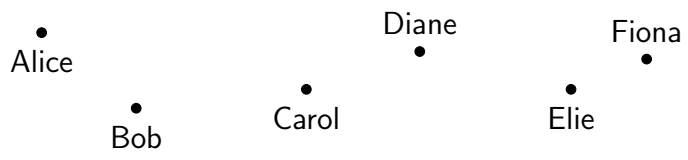
Self-relations can have certain properties or characteristics that are noteworthy. In this chapter, we will introduce three special properties that self-relations can have. These properties are called **reflexivity**, **symmetry**, and **transitivity**. Not every self-relation will have these properties, but some do. Since we will be talking about self-relations in this chapter, I will drop the prefix "self-" and just say "relation," though of course I mean "self-relation."

Remark 20.1. Not every (self-)relation has reflexivity, symmetry, or transitivity properties. Only some of them do. But they are noteworthy when they are present.

20.1 Reflexivity

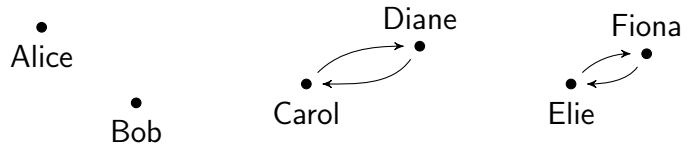
Terminology. A relation on A is **reflexive** if it relates every item in A to itself.

A relation is **reflexive** if it relates every item in the domain to itself. Consider the relation "is the same age as." Suppose we have a set of people:



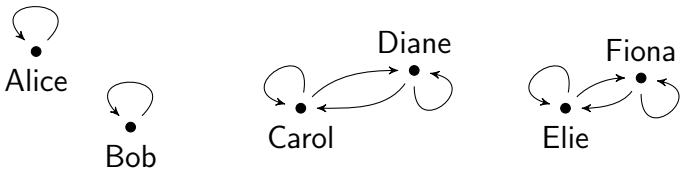
Let's add arrows to show who is the same age as who. Let's say that Diane and Carol are the same age, as are Fiona and Elie:

Remark 20.2. If Diane is the same age as Carol, then Carol is the same age as Diane. Hence, there are two arrows between Carol and Diane, one going each way. Likewise for Fiona and Elie.



But also, each person is the same age *as themself*! So, we need to add self- loops to each dot:

Remark 20.3. Alice is the same age as Alice (herself), Bob is the same age as Bob (himself), Carol is the same age as Carol, and so on for each person in the set.



Remark 20.4. The word "reflexive" means "refers back to oneself." In grammar, a reflexive verb is one whose direct object and subject are the same, e.g., the verb "kicked" in the sentence "John kicked himself."

In this relation, every point is connected to itself. So, if the relation is called R and the set is called A, then this is true:

$$R(x, x) \text{ for every } x \in A$$

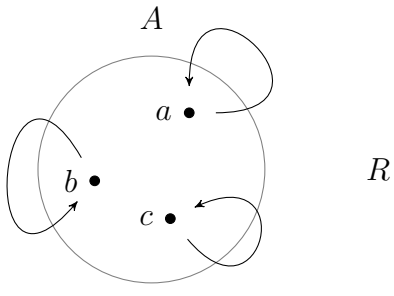
That is to say, for each item x in the set A, the relation R pairs that x up with itself. Such a relation is a *reflexive* relation.

In pictures, what this means is that at each point x, I can travel around a single self-looping arrow to go from x back to x.

We can use this observation to write down a formal definition of a reflexive relation. Let's write it out like this:

Definition 20.1 (Reflexive relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **reflexive** if $R(x, x)$ for every x in A.

Example 20.1. Here is an example of a reflexive relation:



Remark 20.5. Notice that this relation is identical to the **identity function** for A, i.e., id_A, because it maps each item to itself. Every identity function can be seen as an example of a reflexive relation.

We can see here that each item in A is related to itself:

$$R(a, a) \qquad R(b, b) \qquad R(c, c)$$

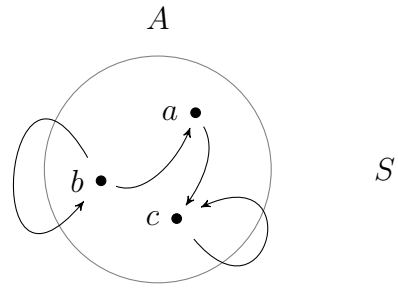
Note, though, that each item has only one arrow coming out of it.

Example 20.2. Here is an example of another reflexive relation, which has more arrows:

Remark 20.6. A reflexive relation can have **other arrows** besides the reflexive ones (self-loops).

Even though this relation has more arrows than the previous one, it is still a reflexive relation, because each item in A has an arrow that loops back to itself.

Example 20.3. Here is a relation that is *not* reflexive:



Remark 20.7. If even a single item in the set is missing a self-loop, then the relation fails to qualify as a reflexive relation.

This is not a reflexive relation because although b and c are related to themselves, a is not related to itself. Hence, this fails to be a reflexive relation.

20.1.1 Reflexivity on a Grid

Recall that we can think of a relation as a **set of pairs**, which we have selected from the **product**. We can see this visually

if we draw the grid, and highlight the pairs in our relation on the grid. For instance, suppose we have this relation:

$$R = \{(0,0), (3,0), (1,2), (2,3)\}$$

We can depict this on a grid by highlighting the pairs:

	0	1	2	3	...
0	**(0, 0)**	(0, 1)	(0, 2)	(0, 3)	...
1	(1, 0)	(1, 1)	**(1, 2)**	(1, 3)	...
2	(2, 0)	(2, 1)	(2, 2)	**(2, 3)**	...
3	**(3, 0)**	(3, 1)	(3, 2)	(3, 3)	...
...	...	...	...	...	...

Reflexive relations are visibly noticeable on a grid, because the **diagonal** will be highlighted. Why is that? Because the diagonal (from top left to bottom right) is precisely the place where every item is paired with itself. We have $(0, 0)$, then $(1, 1)$, then $(2, 2)$, and so on.

Here is a reflexive relation pictured on the same grid. You can see that the diagonal is highlighted:

	0	1	2	3	...
0	**(0, 0)**	(0, 1)	(0, 2)	(0, 3)	...
1	(1, 0)	**(1, 1)**	(1, 2)	(1, 3)	...
2	(2, 0)	(2, 1)	**(2, 2)**	(2, 3)	...
3	(3, 0)	(3, 1)	(3, 2)	**(3, 3)**	...
...	...	...	...	...	...

There can of course be other pairings in a reflexive relation too. For example, other pairings can be highlighted as well:

Remark 20.8. Recall that the **product** of two sets is the set of all possible pairings of elements from those two sets. A product is easily represented as a grid.

Remark 20.9. To depict a relation on a grid, we can highlight just those pairs that are present in the relation at hand.

Remark 20.10. A reflexive relation pairs up **every item** with **itself**. In a grid, the reflexive pairs are on the diagonal, from the top left to the bottom right.

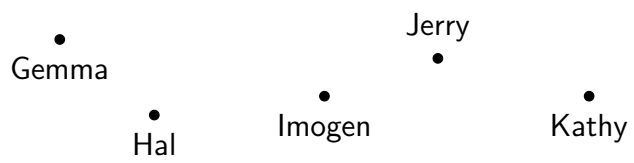
	0	1	2	3	...
0	$(\mathbf{0},\mathbf{0})$	$(0,1)$	$(0,2)$	$(\mathbf{0},\mathbf{3})$	...
1	$(1,0)$	$(\mathbf{1},\mathbf{1})$	$(1,2)$	$(1,3)$	...
2	$(2,0)$	$(2,1)$	$(\mathbf{2},\mathbf{2})$	$(2,3)$	...
3	$(\mathbf{3},\mathbf{0})$	$(3,1)$	$(3,2)$	$(\mathbf{3},\mathbf{3})$	...
...	...	...	...	...	...

Nevertheless, regardless of whether other pairings are present or not, if the relation is a **reflexive** relation, then the **diagonal** will be highlighted.

20.2 Symmetry

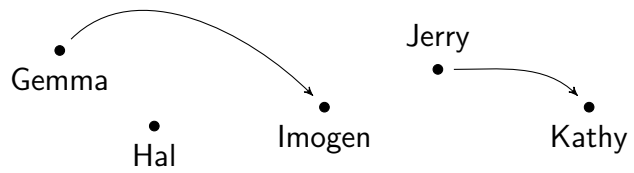
Terminology. A relation on A is **symmetric** if every pairing goes both ways.

A relation R is **symmetric** if each pairing goes both ways. Consider the relation "is a sibling of." Let's take a small set of people:

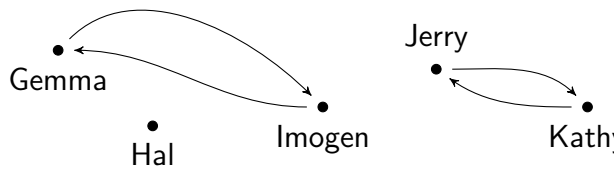


Let's add some arrows to show that, say, Gemma is a sibling of Imogen, and Jerry is a sibling of Kathy:

Remark 20.11. We can see from the picture that Gemma is a sibling of Imogen, and Jerry is a sibling of Kathy. Hal is not a sibling of any of the others in this set.



Of course, to be a sibling goes both ways. If x is a sibling of y, then y is a sibling of x. So we need to add arrows going the other way too:



This relation is **symmetric**, because where there is an arrow going from a point x to a point y, there is *also* an arrow going from a y to x. So, if the relation is called R, then this is true:

$$\text{if } R(x,y) \text{ then } R(y,x), \text{ for every } x,y \in A$$

In pictures, what this means is that if I can go from a point x to a point y by traveling over a single arrow, then I can get from y back to x by traveling on a single arrow too.

We can use this observation to write down a formal definition for symmetric relations. Let's write it out like this:

Definition 20.2 (Symmetric relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **symmetric** if, for any x, y in A, if $R(x,y)$ then $R(y,x)$.

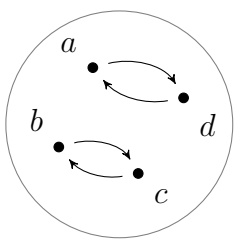
Notice that we say *if* the relation connects x to y, then it must connect y to x. A symmetric relation doesn't have to connect every x to every y in order to be symmetric (e.g,. Hal is not connected to anybody above).

Example 20.4. Here is an example of a symmetric relation:

Remark 20.12. If Gemma is a sibling of Imogen, then Imogen is a sibling of Gemma, so there needs to be two arrows, one for each direction. Likewise for Jerry and Kathy.

Remark 20.13. A symmetric relation doesn't need to connect up every point. The rule here is only that whenever R *does* connect a point x to a point y, then it must *also* connect y to x, if it is to be symmetric.

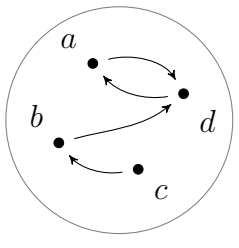
Remark 20.14. Each point x that is connected to a point y also has a connection going the other way. Whenever I can travel across a single arrow to get from x to y, I can also travel across a single arrow to get from y to x. That's what it means to be "symmetric."



We can see in the picture that every time there is an arrow going from an element x to an element y, there is also an arrow going from y to x. Not all dots have arrows between them. For instance, a and b are not connected at all. But, for those arrows that *are* connected, the connection is **symmetric**.

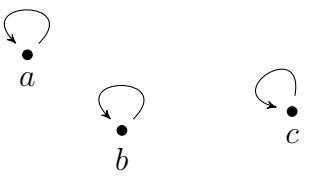
Example 20.5. Here is an example of a relation that is *not* symmetric:

Remark 20.15. Some points may have a two-way link, like a and d in this picture. But in order to qualify as a symmetric relation, *every* pairing that there is must be two-way, and in this picture, some pairings are only one-way (e.g., b to d, or c to b).



This is not symmetric, because I can travel across a single arrow to go from c to b, but not from b to c. Likewise, I can go from b to d, but not from d back to b.

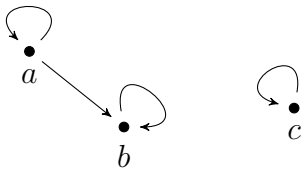
Example 20.6. Here is an example of a relation that, perhaps surprisingly, is symmetric:



Why is this symmetric? The rule says: if I can travel over a single arrow to get from a point x to a point y, then I can also travel over a single arrow to get from y to x. So, what if we ask about a and a? Well, I can get from a to a by traveling around its self-loop, but can I then go the other way (from a back to a)? Of course! I can get from a back to a by following the loop again.

Remark 20.16. Every **self-loop** from a point x to itself is **symmetric**, because you can get from x to x, and then again from x back to x, by following the same loop twice.

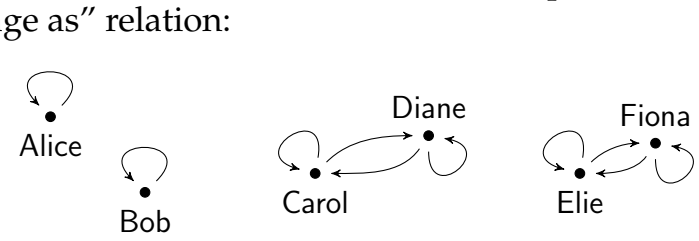
Example 20.7. Here is an example of a relation that is *not* symmetric:



This is not symmetric because I can get from a to b by traveling over a single arrow, but I cannot get from b back to a.

Remark 20.17. In order to show that a relation is *not* symmetric, all you have to do is find **one case** where you can get from a point x to y by traveling over a single arrow, but not back again from y to x.

Example 20.8. Consider the earlier example of the "is the same age as" relation:



In this picture, we can see that every connection between distinct dots goes both ways, and of course self-loops qualify as symmetric too.

20.2.1 Symmetry on a Grid

On a grid, a symmetric relation is easy to spot, because every highlighted pair will be reflected **across the diagonal**. For instance, if $(0, 2)$ is highlighted on the grid, then $(2, 0)$ will be too, and if $(1, 2)$ is highlighted, then $(2, 1)$ will be too.

Remark 20.18. Notice that if you draw a line down the diagonal (from the top left to the bottom right), you will see that every highlighted pair has a mirror image on the other side of that line. If $(0, 2)$ is highlighted on one side of the line, $(2, 0)$ is highlighted on the other side.

	0	1	2	3	...
0	$(0,0)$	$(0,1)$	$\mathbf{(0,2)}$	$(0,3)$	...
1	$(1,0)$	$(1,1)$	$\mathbf{(1,2)}$	$(1,3)$	...
2	$\mathbf{(2,0)}$	$\mathbf{(2,1)}$	$(2,2)$	$\mathbf{(2,3)}$	...
3	$(3,0)$	$(3,1)$	$\mathbf{(3,2)}$	$(3,3)$	...
...	...	...	...	...	...

In this grid, notice how every highlighted pair is reflected across the diagonal. That lets us know that the relation depicted here is symmetric. It reflects a mirror image, so to speak, across the diagonal.

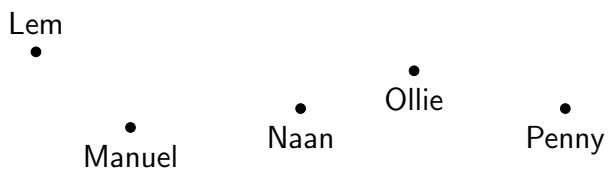
Self-loops are allowed too, since they sit right at the diagonal reflection line. For instance:

Remark 20.19. Self-loops count as symmetric, because the mirror image of (x, x) is just (x, x). For instance, the mirror image of $(2, 2)$ is $(2, 2)$.

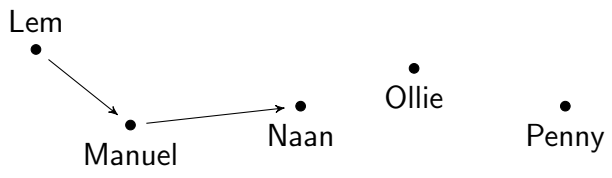
	0	1	2	3	...
0	$\mathbf{(0,0)}$	$(0,1)$	$\mathbf{(0,2)}$	$(0,3)$	...
1	$(1,0)$	$\mathbf{(1,1)}$	$\mathbf{(1,2)}$	$(1,3)$	...
2	$\mathbf{(2,0)}$	$\mathbf{(2,1)}$	$\mathbf{(2,2)}$	$\mathbf{(2,3)}$	...
3	$(3,0)$	$(3,1)$	$\mathbf{(3,2)}$	$(3,3)$	...
...	...	...	...	...	...

20.3 Transitivity

A relation R is **transitive** when this holds true of it: if x is connected to y and y is connected to z, then x is also connected to z. Consider the relation "is shorter than." Suppose we have a small set of people:

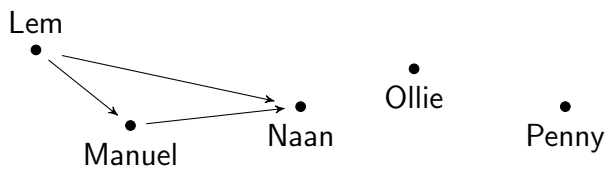


Suppose Lem is shorter than Manuel, and Manuel is shorter than Naan:



Remark 20.20. Notice that we have a chain of two arrows here: we have one link Lem $\mapsto$ Manuel, connected directly to another link Manuel $\mapsto$ Naan.

Well, if Lem is shorter than Manuel, and Manuel is shorter than Naan, then Lem must *also* be shorter than Naan. So we need to add an arrow to show that too:



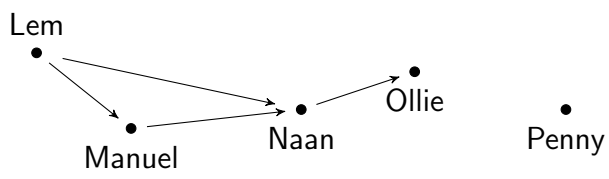
Remark 20.21. Notice that for the two-arrow chain, we connect the start-point up to the end-point. We have Lem $\mapsto$ Manuel and Manuel $\mapsto$ Naan, and then also Lem $\mapsto$ Naan.

This is an example of a **transitive** relation. The basic idea is that the connections propagate up the chain. If Lem is shorter than Manuel, and then Manuel is shorter than Naan, then the connection propagates from Manuel up to Naan too.

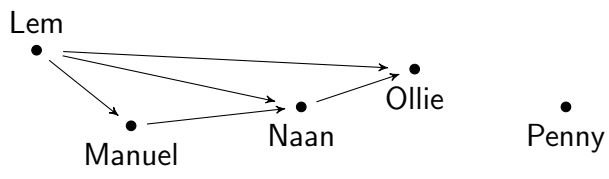
We can put it like this:

if $R(x, y)$ and $R(y, z)$ then $R(x, z)$, for any $x, y, z \in A$

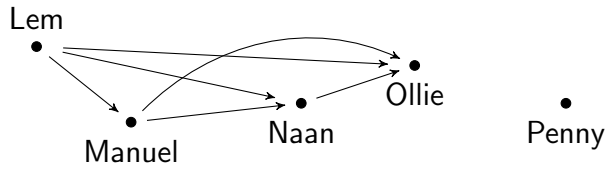
If we add more links onto the end of the chain, it continues to propagate, with more and more arrows. For example, suppose that Naan is shorter than Ollie:



Well, if that is true, then *Lem* is shorter than Ollie too:



And also, *Manuel* is shorter than Ollie:



If you look carefully at this picture, you will notice that for every two-arrow chain, the start-point is also connected up to the end-point. That's what it means to be transitive.

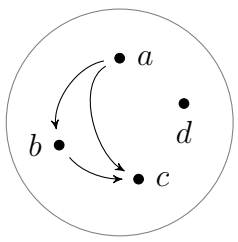
Let's write this down as a definition:

Remark 20.22. Whenever we have a two-arrow chain ($x \mapsto y$ and $y \mapsto z$), the start-point x is also connected up to the end-point z (i.e., $x \mapsto z$).

Remark 20.23. A transitive relation propagates up the chain. So each point lower in the chain needs to be connected to each point later in the chain. In this case, Lem needs to be connected to Manuel, Naan, and Ollie; Manuel needs to be connected to Naan and Ollie; and Naan needs to be connected to Ollie. As an exercise, draw an arrow from Ollie to Penny, and then fill in all the other arrows that are needed to propagate everything up the chain.

Definition 20.3 (Transitive relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **transitive** if, for any x, y, z in A, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.

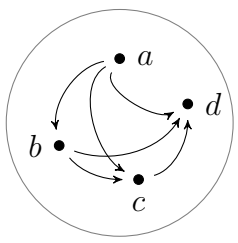
Example 20.9. Here is an example of a symmetric relation:



We can see that there is a two-arrow chain here: a is connected to b, and b is connected to c. We can also see that the start-point of that chain a is connected to the end-point of that chain c. And there are no other two-arrow chains in this picture, so we can conclude that this relation is a transitive relation.

Remark 20.24. There is only one two-arrow chain in this picture: the one made from the arrow from a to b and the arrow from b to c. So we only need to check this one two-arrow chain, to figure out if this relation is transitive. If there were other two-arrow chains in the picture, we would have to check each of them, in order to figure out if the relation is transitive. A relation is transitive only if it connects the start-point and end-point of *every* two-arrow chain.

Example 20.10. Here is another example of a transitive relation:



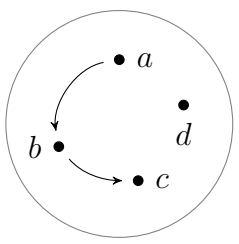
In this relation, we have *four* two-arrow chains:

Remark 20.25. Note that, in this picture, *every* two-arrow chain has another arrow connecting its start-point with its end-point. That is the defining characteristic of a transitive relation.

a to b, b to c $\quad$ b to c, c to d
a to b, b to d $\quad$ a to c, c to d

A transitive relation must connect up the start-point and end-point of every two-arrow chain, and we can see that this does that. It connects a to c for the first chain, it connects b to d for the second chain, and it connects a to d for the third and fourth chains.

Example 20.11. Here is an example of a relation that is *not* transitive:



Here we have a two-arrow chain: a to b, b to c. However, the start-point a is not connected to the end-point c, so this is not a transitive relation.

20.4 Summary

IN THIS CHAPTER, we learned about three properties or characteristics that self-relations can have: reflexivity, symmetry, and transitivity.

- A relation R on a set A is **reflexive** if it relates each item in A to itself. That is, it is reflexive if $R(x, x)$, for every $x \in A$.

- A relation R on a set A is **symmetric** if every pairing it makes in A is two-way. That is, it is symmetric if $R(x, y)$ then $R(y, x)$, for every $x, y \in A$ (even if we pick the same point for x and y).

- A relation R on a set A is **transitive** if it connects up the start- and end-point of every two-arrow chain. That is, it is transitive if, for every $x, y, z \in A$, if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.

[21]

ANTI-PROPERTIES OF RELATIONS

In Chapter 20, we looked at some of the **properties** that certain sorts of self-relations can have. In particular, we noted that a relation can be **reflexive** (if it connects every element in a set to itself), it can be **symmetric** (if, it makes every connection go both ways), and it can be **transitive** (if it connects the start- and end-points of every two-link chain).

In this chapter, we will look at relations that exhibit the opposite properties: namely, relations that are **irreflexive**, **intransitive**, or **asymmetric**/**antisymmetric**.

Remark 21.1. The antithesis of reflexivity is irreflexivity. The antitheses of transitivity is intransitivity. There are two antitheses of symmetry: asymmetry and antisymmetry.

21.1 Irreflexivity

RECALL FROM CHAPTER 20 that a relation is **reflexive** if it connects every item in the domain to itself. So, if the relation is called R and the set is called A, then this will be true:

$$R(x, x), \text{ for every } x \in A$$

Terminology. A relation on A is **reflexive** if every item in A is connected to itself.

By contrast, if a relation connects *no* items to themselves, then it is **irreflexive**. Hence, for an irreflexive relation, this will be true:

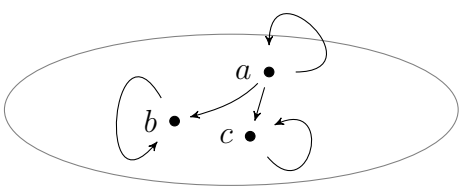
$$\text{for every } x \in A, \text{ it is not the case that } R(x, x)$$

Terminology. A relation on A is **irreflexive** if no items in A are connected to themselves.

We can use this observation to write down a formal definition of an irreflexive relation. Let's write it out like this:

Definition 21.1 (Irreflexive relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **irreflexive** if for every x in A, it is not the case that $R(x, x)$.

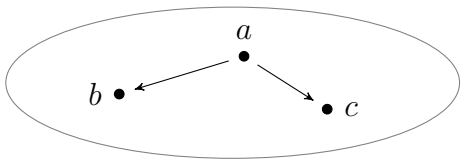
Example 21.1. Here is an example of a reflexive relation:



Remark 21.2. Every point in this picture is connected to itself (each dot has a self-loop). That make this relation **reflexive**.

We can see that this relation is reflexive because each dot has a self-loop (i.e., an arrow looping back to itself).

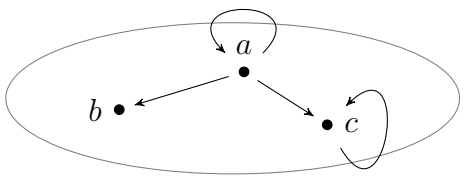
Example 21.2. By contrast, here is an example of an irreflexive relation:



We can see that it is irreflexive because no dot in the picture has a self-loop. Note that other arrows are present in this irreflexive relation. What makes it irreflexive is not whether it has other arrows, but only that it has no self-loops.

Remark 21.3. No point in this picture is connected to itself (no dot has a self-loop). That make this relation **irreflexive**.

Example 21.3. A relation need not be reflexive or irreflexive. A relation can be **neither**. Consider this example:



We can see that two of the dots have a self-loop. In order to be reflexive, *every* dot would need to have a self-loop, and we can see that this is not the case. One dot is missing a self-loop (namely, b), so this relation is not reflexive.

At the same time, in order to be irreflexive, every dot would have to be *missing* a self-loop, and we can see that this is not the case. Two dots have self-loops (namely, a and c), so this relation is not irreflexive either.

Remark 21.4. Not every dot in this picture is connected to itself, but some are. Hence, not every dot has a self-loop, and not every dot is missing a self-loop. That make this relation neither **reflexive** nor **irreflexive**.

21.2 Intransitivity

AS WE SAW IN CHAPTER 20, a relation R is **transitive** if it connects the start-point to the end-point of every two-arrow chain — i.e., if x is connected to y and y is connected to z, then x is also connected to z. We can put it like this:

Terminology. A relation on A is **transitive** when, if x is connected to y and y is connected to z, then x is connected to z too.

$$\text{if } R(x,y) \text{ and } R(y,z) \text{ then } R(x,z), \text{ for any } x,y,z \in A$$

By contrast, a relation is **intransitive** if it does not connect up the start- and end-point of every two-arrow chain. So, if there is at at least one case in R where $R(x,y)$ and $R(y,z)$, but not $R(x,z)$, then R is intransitive. We can put the rule like this. A relation R is intransitive when this is true:

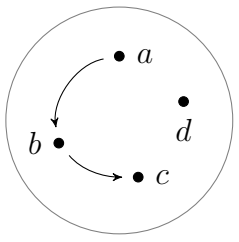
Terminology. A relation on A is **intransitive** when, if there is at least one case where x is connected to y and y is connected to z, but x is not connected to z.

$$\text{if } R(x,y) \text{ and } R(y,z) \text{ yet not } R(x,z), \text{ for some } x,y,z \in A$$

We can use this observation to write down a formal definition for intransitive relations. Let's write it out like this:

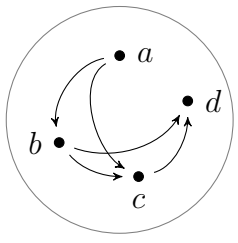
Definition 21.2 (Inransitive relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **intransitive** if, for some x, y, z in A, $R(x,y)$ and $R(y,z)$, but not $R(x,z)$.

Example 21.4. Here is an example of an intransitive relation:



There is a two-arrow chain here: a is connected to b, and b is connected to c. But the start-point a of the chain is not connected to the end-point c of the chain. Hence, this is an intransitive relation.

Example 21.5. With an intransitive relation, some two-arrow chains *can* have their start-points connected to their end-points, so long as there is *at least one* two-arrow chain that is not like this. Consider this example:



In this relation, we have four two-arrow chains:

$$a \mapsto b, b \mapsto c \qquad b \mapsto c, c \mapsto d$$
$$a \mapsto b, b \mapsto d \qquad a \mapsto c, c \mapsto d$$

For two of these chains, the start-point is connected to its end-point:

$$a \mapsto b, b \mapsto c, \text{ and } a \mapsto c \qquad b \mapsto c, c \mapsto d, \text{ and } b \mapsto d$$

But for the other two chains, the start-point is *not* connected its end-point:

$$a \mapsto b, b \mapsto d, \text{ but not } a \mapsto d \qquad a \mapsto c, c \mapsto d, \text{ but not } a \mapsto d$$

Hence, this is an intransitive relation.

Remark 21.5. An intransitive relation does not require that *every* two-arrow chain has its start-point disconnected from its end-point. Some two-arrow chains can have their start-points connected to their end-points, so long as not *every* two-arrow chain is like this.

Remark 21.6. Note that, in this case, *not every* two-arrow chain has an arrow connecting its start-point with its end-point.

Remark 21.7. A relation is either transitive, or it is intransitive. Every relation is one or the other.

Every relation is **either** transitive, **or** it is intransitive. It must be one or the other. For either every two-arrow chain has its start- and end-points connected up (in which case it is transitive), or this is not so for every two-arrow chain (in which case it is intransitive).

21.3 Non-symmetry

Terminology. A relation on A is **symmetric** if every pairing goes both ways.

RECALL FROM CHAPTER 20 that a relation R is **symmetric** if each connection goes both ways. More exactly, if R connects a point x to a point y, it *also* connects y to x:

$$\text{if } R(x,y) \text{ then } R(y,x), \text{ for every } x,y \in A$$

By contrast, a relation is *not* symmetric if this fails to be the case. But, there are two kinds of non-symmetry. One is called **asymmetry**, and the other is called **antisymmetry**. Let's look at each in turn.

21.3.1 Asymmety

Terminology. A relation is **asymmetric** when it is never the case that a point x is connected to a point y and y is also connected to x.

A relation is **asymmetric** if it is never that case that a point x is connected to a point y and also y is connected to x:

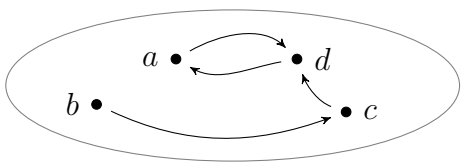
$$\text{if } R(x,y) \text{ then not } R(y,x), \text{ for every } x,y \in A$$

So whenever R connects a point x to a point y, it will *never* connect y to x as well.

We can use this observation to write down a formal definition for asymmetric relations:

Definition 21.3 (Asymmetric relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **asymmetric** if, for every x, y in A, if $R(x, y)$ then it is not the case that $R(y, x)$.

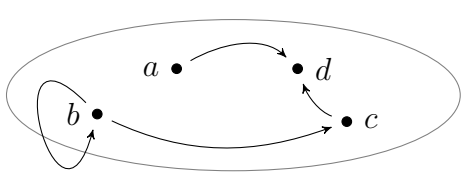
Example 21.6. Here is an example of a relation that is *not* asymmetric:



In order to be asymmetric, if $R(x, y)$, then it cannot be the case that $R(y, x)$. But here, that rule is violated, because $R(a, d)$ and $R(d, a)$.

Remark 21.8. We can see that this relation is not asymmetric, because we can see an arrow going both ways between a and d.

Example 21.7. Here is another example of a relation that is *not* asymmetric:



Why is this not asymmetric? Remember that the rule is this: if $R(x, y)$, then it cannot be the case that $R(y, x)$. That is, if a point x is connected to a point y, then y cannot also be connected to x, no matter which points we pick for x and y. Well, what if we ask about b in the picture?

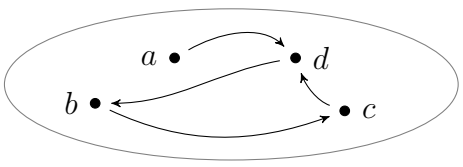
Ponder. None of the arrows look suspicious here, except for the self-loop b. Does b disqualify this relation from being asymmetric?

In the picture, we can see that b is connected to b! Now, if this relation were truly asymmetric, then b could not be connected back to b. But of course, it *is* connected back to b (there's a self-loop). Hence, this relation fails to be asymmetric, because there is a self-loop.

This shows us that there cannot be *any* self-loops in an asymmetric relation. And what does that mean? If there cannot be any self-loops, then it is irreflexive. Hence, an **asymmetric relation** is always **irreflexive** too, because there cannot be any self-loops in asymmetric relations.

An asymmetric relation is truly **one-way**. Every arrow in the picture will go from one point to another, *distinct* point.

Example 21.8. Here is an example of a asymmetric relation:



No two points have an arrow going both ways, and no point has a self-loop. Each arrow goes one way, from one point to a distinct, other point. So this relation is asymmetric.

21.3.2 Antisymmetry

An **asymmetric** relation is one whose arrows always go from a point x to a point y, and no arrow goes back from y to x. This rules out any reflexive self-loops.

A relation is **antisymmetric** if it is asymmetric, but it allows self-loops. So, between any two **distinct** points, there must be one arrow only, going one way. But from any point to **itself**, there can be a self-loop.

Remark 21.9. Recall from Chapter 20 that reflexive self-loops can be seen to go two-way, because you can get from a point x to the point x, and then back again, by following the same self-loop arrow back. Hence, if a relation is **asymmetric**, then it must be **irreflexive** — it cannot have any self-loops. Every arrow must be truly one way: every arrow must go from one point to another (distinct) point.

Remark 21.10. Notice in Example 21.8 that no two points have arrows going both ways, and no points have a self-loop. If there is an arrow, it goes one way only, from one point to another, different point. That is what it means to be asymmetric.

Terminology. An **antisymmetric** relation is like an asymmetric relation, except it allows self-loops. Between any two distinct points, there can be only one arrow, going one way only. But there can also be self-loops.

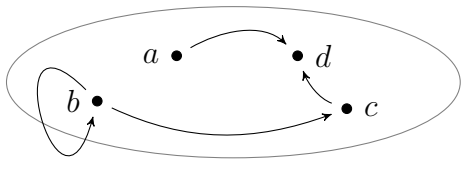
Since we allow self-loops, we can say that x is connected to y and y is connected to x when x and y are the same point. We can put the idea like this:

$$\text{if } R(x,y) \text{ and } R(y,x) \text{ then } x = y, \text{ for every } x, y \in A$$

We can use this observation to write down a formal definition for antisymmetric relations:

Definition 21.4 (Antisymmetric relations)**.** For any set A and relation $R \subseteq A \times A$, we will say that R is **antisymmetric** if, for every x, y in A, if $R(x,y)$ and $R(y,x)$, then $x = y$.

Example 21.9. Here is an example of a relation that is antisymmetric:



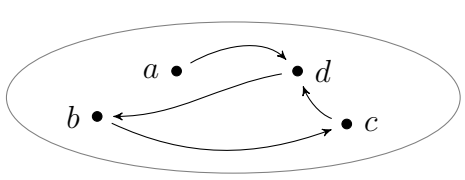
In order to be antisymmetric, if $R(x,y)$, then it cannot be the case that $R(y,x)$, unless $x = y$. Here, that is true. No two points are such that $R(x,y)$ and $R(y,x)$, except for b and b, which are of course the same point.

Visually, we can see this. There are no two way connections between distinct points, but there is a self-loop.

Remark 21.11. The defining mark of an antisymmetric relation is that the only cases where a point x is connected to a point y and also y is connected to x are cases where x and y are the same point, i.e., when $x = y$. If x and y are different points, then there can only be one arrow between them, going one way only (so x will be connected to y, or y will be connected to x, or there simply won't be an arrow between them at all).

Remark 21.12. We can see that this relation is antisymmetric, because there are no two-way arrows between different points, but there is a self-loop.

Example 21.10. Here is another example of an antisymmetric relation:



Notice that this relation is **asymmetric**, because there are no two-way arrows, and there are no self-loops. But such a relation is *also* **antisymmetric**, because it satisfies the definition of antisymmetry. An antisymmetric relation allows self-loops, but it does not require that they be present.

Remark 21.13. Every **asymmetric** relation is **antisymmetric**, but not vice versa. In particular, if it is not symmetric but there are self-loops, then it is *anti*-symmetric, and not *a*-symmetric.

This tells us something important: every **asymmetric** relation is an **antisymmetric** relation, but not vice versa. An asymmetric relation is just an antisymmetric relation with no self-loops. An antisymmetric relation with self-loops, on the other hand, is not asymmetric. When self-loops are present, that pushes it out of pure a-symmetry land (the non-symmetry land without self-loops), and lands it in anti-symmetry land (the non-symmetry land with self-loops).

21.4 Summary

IN THIS CHAPTER, we learned about some anti-properties of relations: irreflexivity, intransitivity, and non-symmetry.

- A relation R on a set A is **irreflexive** if it relates no item in A to itself. That is, it is irreflexive if for every $x \in A$, it is not the case that $R(x, x)$.
- A relation R on a set A is **intransitive** if it fails to connect up the start- and end-point of at least one two-arrow chain. That is, it is intransitive if, for some $x, y, z \in A$, $R(x, y)$ and $R(y, z)$ but not $R(x, z)$.

- There are two kinds of non-symmetry: **asymmetry** and **antisymmetry**.
- A relation R on a set A is **asymmetric** if, for any $x, y \in A$, if $R(x, y)$ then it is not the case that $R(y, x)$. This excludes reflexivity (there cannot be any self-loops in an asymmetric relation).
- A relation R on a set A is **antisymmetric** if, for any $x, y \in A$, if $R(x, y)$ and $R(y, x)$, then $x = y$. In other words, the only way that a point x can be connected to a point y and y can be connected to x is when x and y are the same point. So antisymmetric relations are like asymmetric relations, but they allow self-loops.
- Every **asymmetric** relation is **antisymmetric**, but **not vice-versa**.

[22]

STRUCTURES

WE HAVE LOOKED AT sets, functions, and relations. **Sets** are collections of items, **functions** map each item in one set with an item in another (possibly the same) set, and **relations** are any pairing of elements between two sets (possibly the same set).

These are the basic building blocks that mathematicians use to build what are called mathematical **structures**. In this chapter, we will introduce the concept of structures.

22.1 Building Structures

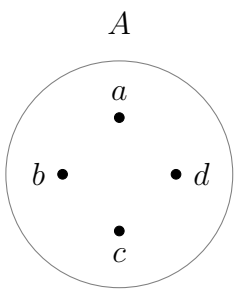
TO CONSTRUCT a **structure**, we take one or more sets, and we add to them whatever functions and relations we like. The

Terminology. A **structure** is one or more sets equipped with any number of functions and relations. The functions and relations give the sets extra structure.

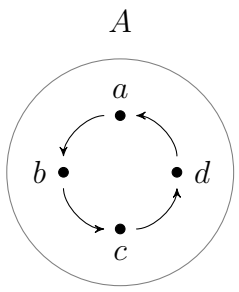
functions and relations give the sets extra structure. That's all a structure really is. It is a set (or a bundle of sets), equipped with some functions and relations.

Recall that a set by itself has no *internal* structure. It is just a bag of items, with no order or repetition inside of it. Functions and relations can *add* structure to sets, by pairing up elements inside them. Adding a function or relation lets us say, "these two elements have a line between them, and those two elements have a line between them," and so on.

As a simple visual example, suppose we have a set A:



Now suppose we add a function $f : A \to A$, e.g.:

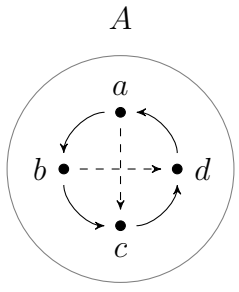


equipped with f

We can see that just by adding this one function, we add a lot of structure to the set. Suppose we add a relation $R \subseteq A \times A$ as well (I'll draw it with dashed arrows):

Remark 22.1. If you like, you can think of the *sets* as big bags full of nails, screws, boards, and other building materials. There is no structure inside these bags. They're just unorganized bags of items. Adding *functions* and *relations* to the sets is like taking pieces from the bags, and putting them together, in some particular way. The pieces in the bags go from being loose items, to standing in structured relationships with each other.

Remark 22.2. The function $f : A \to A$ pictured here consists of the following mappings: $f(a) = b$, $f(b) = c$, $f(c) = d$, and $f(d) = a$.



equipped with f, R

We can see that we have added even more structure now. So this is the basic idea behind adding structure to sets. We start with one or more sets, and we add functions and relations in order to endow the sets with whatever structure we need.

There are no restrictions on which sets, functions, or relations we want to use, to build up a structure. We can pick whichever sets, functions, and relations we like. We can start with one, two, or 15 sets. We can add one, two, or 10 functions. We can add one, two, or 20 relations. We can just add functions and no relations, or we can just add relations and no functions. It is really up to us.

Remark 22.3. The relation $R \subseteq A \times A$ pictured here consists of the following mappings: $R(a, c)$, and $R(b, d)$.

Remark 22.4. We can add whatever functions and/or relations we like to sets, to give them the structure we need.

22.2 Definition and Notation

Once we have constructed a structure, we need a way to write it down, so that we can communicate about it with our peers. Like most mathematical entities, we can give names to our structures too. Let's use bolded capital letters. E.g., $\mathbf{S}$, $\mathbf{T}$, and so on.

We will refer to the sets as the **base sets** or synonymously, as the **carrier sets**. The idea here is simply that the sets are the "bases" on which we build the structure. It is the sets that "carry" the structure.

To denote a structure, we will take the base sets, and whatever functions and relations we have equipped them with, and we will put them all into a **tuple**. We will list the sets first, then the functions, and then the relations. Here is the basic format:

$$\mathbf{S} = (A_1, A_2, \ldots, f_1, f_2, \ldots, R_1, R_2, \ldots)$$

This is just a template. I have written $A_1, A_2, \ldots$ to indicate that there might be any number of sets in a structure, but you should replace that with whichever sets you like. Likewise for the functions $f_1, f_2, \ldots$, and the relations $R_1, R_2, \ldots$

A moment ago, we drew a picture of a simple structure by taking one base set A, and equipping it with a function f and a relation R. Using our template, we can denote such a structure like this:

$$(A, f, R)$$

Note that we are following the template: we list the sets first, then the functions, then the relations. In this case, there is just one set (namely, A), so that goes first, then there is one function (namely, f), so that goes second, and there is just one relation (namely, R), so that goes last.

We can name this structure. Let's pick $\mathbf{S}$. Hence, we can write this:

$$\mathbf{S} = (A, f, R)$$

Read that alound like so: "$\mathbf{S}$ is a structure consisting of the tuple (A, f, R)." You could even read it like this: "$\mathbf{S}$ is a structure consisting of a base set A, equipped with a function f and a relation R."

Of course, your readers need to know what exactly A, f, and R consist of, so if you have not already specified what

Remark 22.5. Recall from Section 11.5 that a **tuple** is an ordered list of items surrounded by round braces. A pair (x, y) is a 2-tuple, a triple (x, y, z) is a 3-tuple, a quadruple (x, y, z, w) is a 4-tuple, and so on.

Notation 22.1. We will use **bolded capital letters** as names for structures, and we will specify the contents of the structure with **tuples**, where we list the base sets first, then the functions, then the relations.

those are, you need to do so. It is common in mathematical writing to specify a structure in the following way:

> Let $\mathbf{S} = (A, f, R)$ be a structure where:
>
> - $A = \{a, b, c, d\}$
> - $f : A \to A$ is: $f(a) = b$, $f(b) = c$, $f(c) = d$, and $f(d) = a$
> - $R \subseteq A \times A$ is: $R(a, c)$, and $R(b, d)$

Read that aloud like so: "let $\mathbf{S}$ be a structure consisting of the tuple (A, f, R), where A is the set $\{a, b, c, d\}$, f is a function from A to A such that $f(a) = b$, $f(b) = c$, $f(c) = d$, and $f(d) = a$, and R is a relation from A to A such that $R(a, c)$, and $R(b, d)$."

Notice that, in this case, the relations and functions we use to build this structure are **self-functions** and **self-relations**: they are mappings from A to itself. We are not dealing with mappings from one set to *another* set here. We are dealing with mappings that connects elements in A to other elements in the same set.

But this makes sense. We have only one base set in this example, and so any structure we want to add to it will be structure that we add to the **insides of the set**. So of course the functions and relations we have in this structure are functions and relations that pair up elements in the set with other elements *in the same set*.

If our structure had more base sets (for example, if it had A and B as base sets), then we could add functions and relations that go from A to B, B to A, A to A, or B to B. With more base sets, we can add structure not only inside the sets, but also between the sets.

Let us take all that we have said here, and let us write it down as a definition for structures.

Remark 22.6. You will notice that this description of a structure is quite hard to understand, just by reading it. If you are reading someone else's math writing, and they specify a structure in this fashion, it is important to pull out a piece of scrap paper, and try to work out on paper what the structure really amounts to. Draw some pictures (like we did above), and scribble out notes for yourself. It takes this kind of active work to fully internalize the idea.

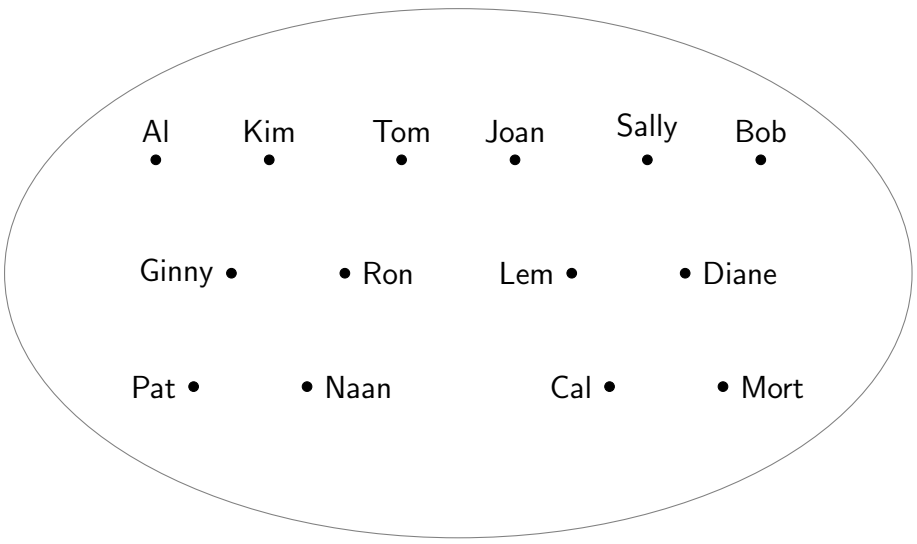
Remark 22.7. When we build a structure with one base set (call it A), the functions and relations that we add to it will go from A to A. This is because we are adding structure *inside* the set A. If we build a structure with more than one base set, then we can add relations that add structure both *inside* the sets, but also *between* the sets.

Definition 22.1 (Structures)**.** We will say that a **structure** consists of one or more sets A_1, A_2, ... equipped with zero or more functions $f_1, f_2, \ldots$ on those sets, and zero or more relations $R_1, R_2, \ldots$ on those sets. We will call the sets the **base sets** or the **carrier sets** of the structure. We will use a bolded capital letters (such as $\mathbf{S}$ or $\mathbf{T}$) as **names** for structures, and we will denote a structure as a tuple with this shape: $\mathbf{S} = (A_1, A_2, \ldots, f_1, f_2, \ldots, R_1, R_2, \ldots)$.

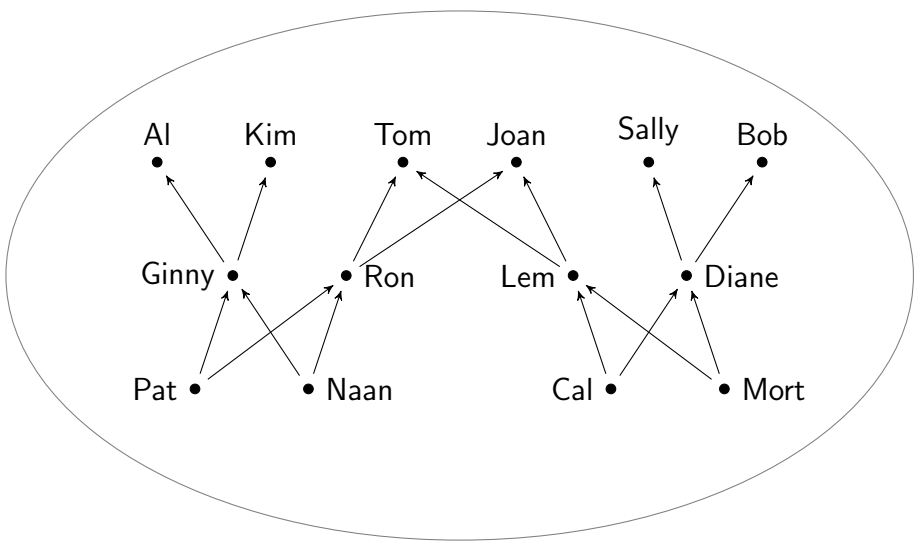
22.3 Example

Let's construct an example that models the structure of a small company. Let's start with a base set A of employees:

Remark 22.8. We will start with a base set, as an oval filled with some labeled dots. Then we will add relations, which we will depict with arrows. As we proceed, it would be a good exercise to try and write down what the base set is, in **set-roster notation**. Then try to write down what each relation is, as a **list of pairs**.

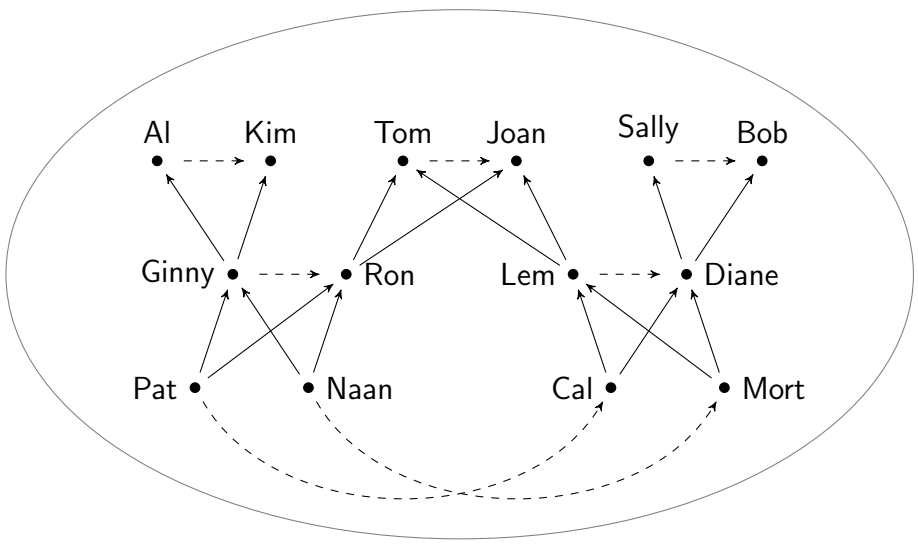


Next, let's add a relation W that indicates who works on projects for whom. Let's suppose it looks like this:



So, Ginny works on projects for Al and Kim, Ron works on projects for Tom and Joan, and so on.

Finally, let's add a relation called M that indicates a mentoring relationship. Let's draw this relation with dashed arrows, and it indicates which employees are assigned as mentors to which others:



Remark 22.9. As we add more lines to our picture, we can see more of the structure that we are adding to our base set. However, the picture need not look exactly like this. In fact, we can draw the dots anywhere on the page, so long as the arrows stay fixed between the same dots. Try drawing this same structure, but lay out the dots on your paper in a different way than the way they are arranged here. Make sure you connect up the dots in the same way though. The arrows in both pictures should connect up the same dots.

We can see from the picture that Ginny mentors Ron, Pat mentors Cal, Sally mentors Bob, and so on.

All in all then, we have a structure here. Our base set A is the set of employees, but we've enriched our set with a relation W designating who works on projects for whom, and a relation M which designates who mentors who.

Notice that this structure is equipped with no functions. It is constructed only from a base set and relations. This is okay. As we said before, we can build our structures with whatever functions or relations we need, that might suit our purposes. In this case, we didn't need any functions to model the structure of our little company. All we needed were relations, and so we built our structure from just those relations.

Let's call our structure $\mathbf{S}$, and let's write it all out fully:

Let $\mathbf{S} = (A, W, M)$ be a structure that models our little company, where:

- A is the set of employees: { Al, Pat, Kim, Tom, Joan, Sally, Bob, Ron, Ginny, Lem, Naan, Mort, Cal, Diane }
- W is a relation donating who works on projects for whom: { (Ginny, Al), (Ginny, Kim), (Ron, Tom), (Ron, Joan), (Lem, Tom), (Lem, Joan), (Diane, Sally), (Diane, Bob), (Pat, Ginny), (Pat, Ron), (Naan, Ginny), (Naan, Ron), (Cal, Lem), (Cal, Diane), (Mort, Lem), (Mort, Diane) }
- M is a relation denoting who mentors who: { (Al, Kim), (Tom, Joan), (Sally, Bob), (Ginny, Ron), (Lem, Diane), (Pat, Cal), (Naan, Mort) }

Remark 22.10. Neither W nor M are functions. Can you see why they fail to be functions? W fails to be a function because a number of points have more than one solid arrow coming out of them. M fails to be a function because there are a number of points that don't have a dashed arrow coming out of them.

Remark 22.11. It is a good exercise to "translate" between the pictures and the formal descriptions of structures. Take one of the pictures of a structure we gave above, and see if you can write it down as a formal description. Then take a formal description of a structure, and see if you can draw a picture.

As an exercise, see if you can take this formal description of our structure, and draw a picture of it on scratch paper. Try and make your picture look different from the one we

drew above. You can put your dots anywhere on the page. The important thing is that, wherever you draw the points on the page, the arrows connect precisely the points that are specified in our description here.

22.4 Empty Structures

What if we have a structure that has one base set, and it is equipped with zero functions and zero relations. For instance, suppose we have a base set A, with nothing more:

$$\mathbf{S} = (A)$$

Notice that there are no fs or Rs after the A. This structure is just a 1-tuple, which contains nothing but the base set A.

Remark 22.12. We have 2-tuples, 3-tuples, 4-tuples, and so on, but we can also have a 1-tuple.

Is this a structure? We can say that it is a structure, but it is entirely **empty** of structure. It is like an abandoned building site where all of the construction materials are lying around, and nobody put any of them together.

Such a structure is basically equivalent to the set A, since the set A is a mere set, and like any other set, it has no internal structure. Only when we add functions and relations to the structure do we end up adding any internal structure to the base set.

22.5 Summary

IN THIS CHAPTER, we learned about structures, and we learned how we build them by taking base sets and equipping them with functions and relations so as to add structure to the base sets.

- A **structure** consists of one or more sets equipped with zero or more functions on the sets, and zero or more relations on the sets.
- We call the sets the **base sets** or the **carrier sets**.
- We will use **bolded capital letters** like $\mathbf{S}$ or $\mathbf{T}$ for the **names** of structures.
- We will denote a structure as a **tuple**, where we list the base sets first, then the functions, and then the relations, using this template: $\mathbf{S} = (A_1, A_2, \ldots, f_1, f_2, \ldots, R_1, R_2, \ldots)$.
- If we have not specified what the base sets, functions, and relations actually consist in, then when we specify a structure, we need to be sure to **lay it all out clearly** for the reader, so they know exactly what the structure we have in mind consists of.

[23]

FURTHER READING

To pursue relations further, the following list may offer some helpful starting points.

- Stewart and Tall (2015, ch. 4) provides a good but introductory-level discussion to the basic concept of relations.
- Steinhart (2018, ch. 2) also provides an excellent introduction for the beginner to the topic of relations.
- Cummings (2020, ch. 9) presents a slow, thorough discussion of relations and how to construct proofs about them.
- Zach (2019, ch. 2) offers a nice, introductory (yet still technical) discussion of relations.
- Velleman (2019, ch. 4) offers an excellent and thorough introduction to the concept of relations, along with a thor-

ough discussion of proofs about them.

- Warner (2019, ch. 3) provides a thorough discussion of relations, with fully-worked proofs.
- Madden and Aubrey (2017, ch. 10) covers relations in detail, along with a thorough discussion of proofs about them.
- Jongsma (2019, ch. 6) provides a rigorous introduction to many of the basic theorems and proofs involved with relations.
- Pinter (2014, ch. 3) offers a simple discussion of the basic theorems and proofs for relations.

Part [5]

GRAPHS

There are many situations that involve some kind of a **network**. For instance, we can think of the U.S. interstate system as a network of roads between cities. We can think of a social network as a network of friends. In all of these scenarios, there is a common structure: we have a set of objects (cities, people, or whatever), and a set of connections between them (roads, being-acquainted-with, etc). Mathematicians call these structures **graphs**, and they call the study of graphs **graph theory**.

In Part 5, we discuss **graph** and **graph theory**. We look at what graphs are, how they are defined, and some of their properties. We also look at **graph isomorphisms**, which are maps between graphs that are essentially the same because they have same shape.

[24]

NETWORKS

THERE ARE MANY SCENARIOS that involve a **network** of connections between objects. For instance, the internet is a network (of connections between routers), and the U.S. interstate system is a network (of freeways between cities).

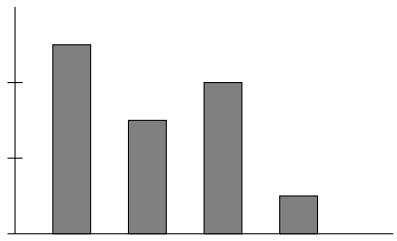
Mathematicians call such networks **graphs**, and the study of networks is called **graph theory**. In this chapter, we will look at the basic structure of graphs.

Ponder. What other networks do you engage with on a daily basis? Perhaps project managers at work? A chain of command in the military? Electrical circuits in a vehicle?

24.1 Plots and Graphs

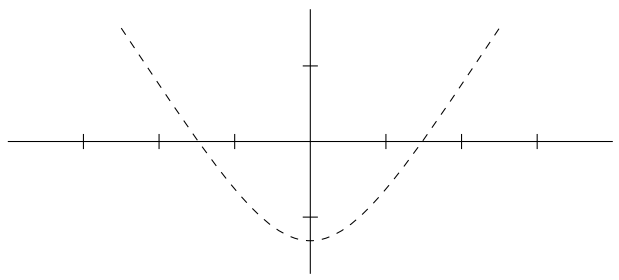
WHEN WE HEAR THE WORD "GRAPH," we might think of various kinds of data visualizations, or even drawings we did on graph paper in school.

For example, as an example of a "graph," we might think of a data visualization such as a bar plot, e.g., something that looks like this:



Remark 24.1. Sometimes visualizations with bars are called "bar graphs," but to avoid mixing up our terminology, we will call them "bar *plots*."

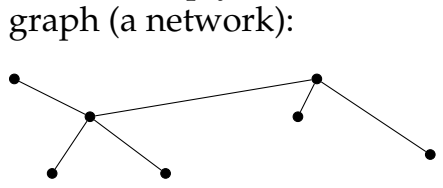
Alternatively, as an example of a "graph," we might think of plotting a curve on graph paper, for instance:



Remark 24.2. Sometimes this is called the "graph" of a curve, but to avoid mixing up our terminology, we will call this a "plot" of the curve.

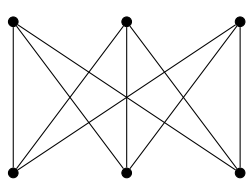
These sorts of drawings are sometimes called "graphs," but to avoid mixing up our terminology, let's call these **plots** instead. In the context of graph theory, a plot is not what a mathematician has in mind when they speak of a "graph."

In the context of graph theory, a **graph** is really a **network**. We can usually draw it simply with dots and lines. Here is an example of a graph (a network):



Terminology. In the context of graph theory, a graph is not a **plot**. Rather, a graph is always a **network** comprised of points that are connected in some way. In graph theory, the term "graph" is just a synonym for "network."

Here is another graph (i.e., a network):



As you can see, these "graphs" have little relation to the plots we mentioned earlier. In graph theory, when we speak of graphs, we are always speaking of networks.

24.2 Definition and Notation

CONCEPTUALLY SPEAKING, a graph is just a set of objects, along with some pairings of those objects. That's really all a network is. You have some *things* (some objects, whatever they might be), and then you have some *connections* between them (pairings of objects).

Here is an example. Suppose we have four people (these are the objects we care about):

$$\{\mathsf{Alice}, \mathsf{Bob}, \mathsf{Clara}, \mathsf{Dan}\}$$

Next, suppose some of these people know each other. Suppose, for example, that Alice and Bob know each other, Clara and Dan know each other, and Bob and Clara know each other too. We can put these down as pairs:

$$\{(\mathsf{Alice}, \mathsf{Bob}), (\mathsf{Clara}, \mathsf{Dan}), (\mathsf{Bob}, \mathsf{Clara})\}$$

So we have a set of objects, and a set of pairings of those objects. Let's give each of these two sets names. Let's call the first set V. Hence:

$$V = \{\mathsf{Alice}, \mathsf{Bob}, \mathsf{Clara}, \mathsf{Dan}\}$$

Remark 24.3. A network consists of some **objects**, and some **pairings** of those objects. We have some things (the objects), which are connected (paired up).

Remark 24.4. The **objects** in our network are these four people.

Remark 24.5. The **connections** in our network are these pairings. There are three connections.

Terminology. We choose "V" for **vertices**, which is what graph theorists call the objects or "points" in a graph. We'll come back to this name below.

Let's call the second set E:

$$E = \{(\mathsf{Alice}, \mathsf{Bob}), (\mathsf{Clara}, \mathsf{Dan}), (\mathsf{Bob}, \mathsf{Clara})\}$$

Now, let's put the two together, and say that our graph is the **tuple** of V and E:

$$(V, E)$$

Read that aloud like this: "the tuple consisting of the set V and the set E."

Let's give this graph a name. Let's call it $\mathbf{G}$. Hence, we can specify our graph by writing this:

$$\mathbf{G} = (V, E)$$

Read that aloud like this: "the graph $\mathbf{G}$ is the tuple consisting of the set V and the set E." Or, if you like, you could read it like this: "the graph $\mathbf{G}$ is defined as the tuple (V, E)."

To draw this graph, we draw a dot for each object in V. We can put the dots wherever we like on the page. For instance, we might arrange them on the page like this:

Next, we draw a line to represent each pairing in E. So, for example, since the pair $(\mathsf{Alice}, \mathsf{Bob})$ is in E, we can draw a line between their two dots:

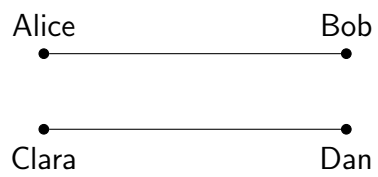
Terminology.We choose "E" for **edges**, which is what graph theorists call the connections or "lines" in a graph. We'll come back to this below too.

Remark 24.6. Recall from Chapter 11 that a **tuple** is an ordered sequence of items. In this case, a 2-tuple (also called a "pair") is a sequence of two items: in the first slot we have V, and in the second slot we have E.

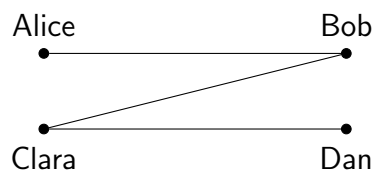
Remark 24.7. Here we arranged the four dots in a rectangular fashion, but this is not necessary. We could put the dots anywhere on the page.

Remark 24.8. If we initially drew our dots in a different arrangement, no matter where we put those dots, we'd want to connect up the dot labeled "Alice" with the dot labeled "Bob."

Clara and Dan are also paired up, so we add a line for their connection:



And Bob and Clara are also paired up, so we add a line for them too:



With that, we have drawn a picture of our graph. You can see that it visually represents the sets V and E exactly, because it has a dot for each object in V, and a connecting line for each pairing in E.

In graph theory, we often call the objects in V the **vertices** of the graph, because a **vertex** is a point where lines come together. We call the pairings in E the **edges** of the graph, because in geometry, we often refer to lines as "edges" (as in, the edge of a shape).

Let's take what we have said and put down a definition for a graph.

Definition 24.1. We will say that a **graph** is a tuple (V, E), where V is a set of objects, and E is a set of pairs of objects from V. We will call V the **vertices** of the graph, and we will call E the **edges** of the graph. We will use bolded, capital letters as names for graphs, e.g., $\mathbf{G}$, $\mathbf{H}$, etc.

Remark 24.9. Again, it doesn't matter if we initially drew our dots in a different arrangement than what we have pictured here. No matter where we put those dots, we'd want to connect the same dots we are connecting here (i.e., we'd want to connect "Clara" and "Dan," as well as "Bob" and "Clara."

Terminology. The word **vertices** is plural (many of them), and **vertex** is singular (one of them). Since graph theorists call the objects **vertices** and the connections **edges**, we will often name the set of objects in our graphs "V" and the pairings "E." That's why we chose those names above.

Notice how we really have captured the **essence** of a **network** here. If we strip away inessential details, a network is nothing more than a bunch of objects that are connected. For instance, a set of cities and the roads between them can be seen as just a set of objects (in this case, cities) and connections (roads). A social network can be seen as a set of objects (people) and connections (who knows who). That is exactly what we've defined a graph to be.

24.3 Different Drawings

A GRAPH IS ACTUALLY not the same thing as its drawing. A drawing of a graph is really just a *picture* or *representation* of a graph.

Remark 24.10. A **graph** is not identical to its **drawing**. There are many drawings of a single graph. Each drawing is just a **representation** of a graph.

Exactly where we draw the dots in a picture doesn't matter. What matters is only that the points are connected to each other in the right way. For example, let's draw the above graph G in a different way. Let's draw the four dots in different places:

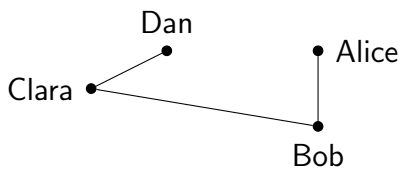
Remark 24.11. Notice how the dots are not arranged in a triangular fashion as before, and they are not even in the same order (going clockwise around).

Now let's draw in the connections. We have Alice connected to Bob:

Remark 24.12. Notice that in this picture and the picture we drew above, there is a line between Alice and Bob.

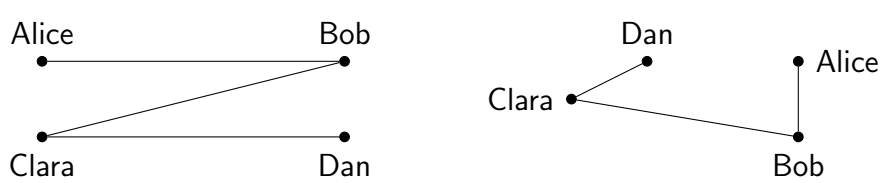
Then Clara connected to Dan:

And finally, Bob connected to Clara:



Ponder. Compare this drawing to the earlier drawing of the same graph. What's different about it? And what's the same?

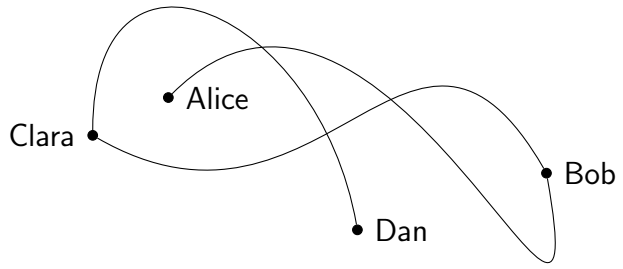
Notice that this picture looks different from the one we drew before. Here they are side by side:



Remark 24.13. Notice that when we look at this drawing and compare it to the earlier drawing, the same points are connected, even though we have drawn the points in different places on the page. Regardless of how the points are positioned in these two drawings, there is still the same number of connections, between the same points.

But these depict the very **same graph**. We know that because they have exactly the same *points* (Alice, Bob, Clara, and Dan), and they have the same *connections*: in both pictures, Alice is connected to Bob, Clara is connected to Dan, and Bob is connected to Clara.

What this shows us is that the essence of a graph is not the picture of it. Rather, it is the **points** (the objects), and the **connections** between them (the pairings). We could draw the lines curvy if we liked, so long as we connect up the right points, for instance like this:



24.4 Summary

IN THIS CHAPTER, we learned about the basic structure of graphs (which are networks, not plots).

- A **graph** G is defined as a tuple (V, E), where V is a set of objects, and E is a set of pairings of those objects. We call V the **vertices** of the graph, and we call E the **edges** of the graph.
- When we draw a graph, we represent the vertices as points, and we draw the connections as lines.
- A **graph** and **its drawings** are not the same. There can be many different drawings of the same graph. Two drawings are drawings of the same graph if they have the same vertices and the same edges.

[25]

PROPERTIES OF GRAPHS

IN CHAPTER 24 we discussed **graphs**, which are network structures comprised of points connected by lines. The study of graphs is called **graph theory**.

In this chapter, we will look at some of the basic properties of graphs, and we will look at certain kinds of graphs that are discussed so often that they have special names.

Ponder. What are some of the characteristics of a graph that you might write down if you were asked to describe it? The number of vertices? The number of edges? How many edges touch a vertex?

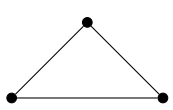
25.1 Properties of Graphs

GRAPH THEORISTS HAVE COME UP WITH some special names to refer to various properties or characteristics of graphs. In this section, we will cover some of these properties.

25.1.1 Order

Terminology. Recall from Chapter 8 that the size of a set is called its **cardinality**. So, the **order** of a graph is the **cardinality** of its set of vertices V, which we denote as $|V|$.

The **order** of a graph is the number of vertices it has. For example, consider this graph:

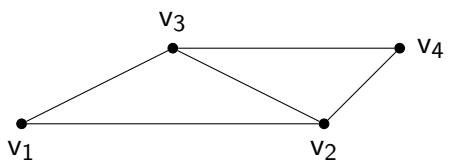


That graph has order 3, because it has three vertices.

25.1.2 Degree

Terminology. The **degree** of a vertex is the number of edges connected to it.

The **degree** of a vertex is the number of edges connected to it. For example, consider this graph:

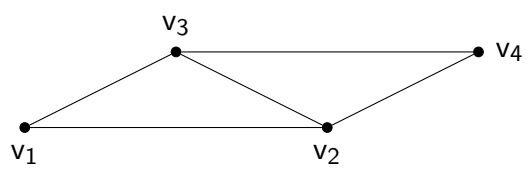


In this graph, vertex v_1 has a degree of 2, because there are two edges connected to it. Vertex v_2 has a degree of 3, v_3 has a degree of 3, and v_4 has a degree of 2.

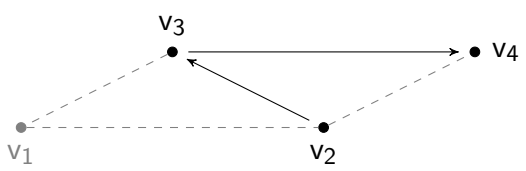
25.1.3 Path

Terminology. A **path** through a graph is a route from one vertex to another vertex, by traveling through intermediate vertices between. Some math books call a path a **walk**.

A **path** through a graph is a walk from one vertex in the graph to another vertex in the graph, by traveling from vertex to vertex along edges between them. For example, consider this graph:

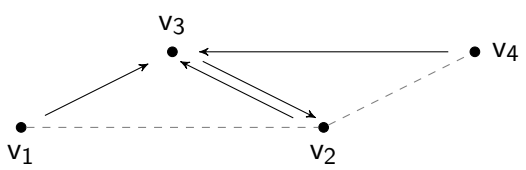


The next picture shows a path from v_2 to v_4 (the path is indicated with arrows to show each step of the journey):



The path starts at vertex v_2, then it goes to vertex v_3, then it goes to vertex v_4. We can write this concisely like this: v_2-v_3-v_4.

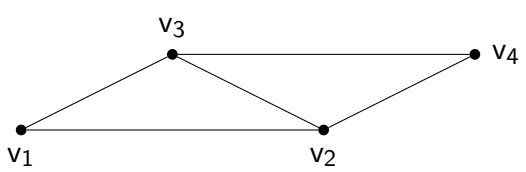
Here is another path, v_4-v_3-v_2-v_3-v_1:



Remark 25.1. There are two legs to this journey. The first leg is from v_2 to v_3. The second is from v_3 to v_4.

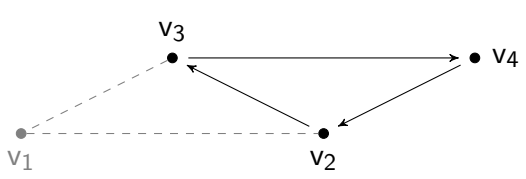
25.1.4 Cycle

A **cycle** through a graph is a path that starts and ends at the same point. Consider this graph again:



Terminology. A **cycle** through a graph is a path that starts and ends at the same vertex. Some math books call a cycle a **circuit**.

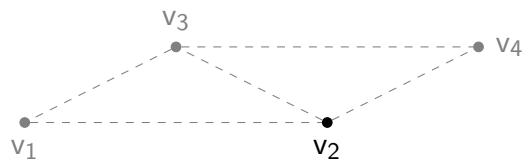
The next picture shows a cycle from v_2 to v_2:



Remark 25.2. This path starts at v_2, and it ends at v_2. Since it starts and ends at the same point, it's a cycle.

The path starts at v_2, it then goes to v_3, then to v_4, and finally, it goes back to v_2 again. Since this path starts and ends at the same point, it is a **cycle**.

Now consider this picture:

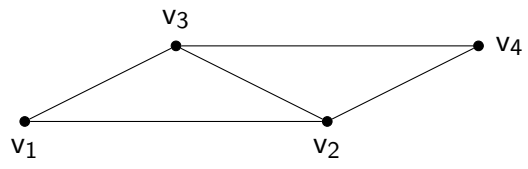


Remark 25.3. This is an example of a **trivial cycle**. If you're already home, how do you get back home? One way is just to not leave the house.

This is actually a cycle. It's just that it's the shortest possible cycle. We start at v_2, and we end up at v_2, by simply not going anywhere! This cycle has a distance of zero, so to speak. We can call such a cycle a **trivial cycle**.

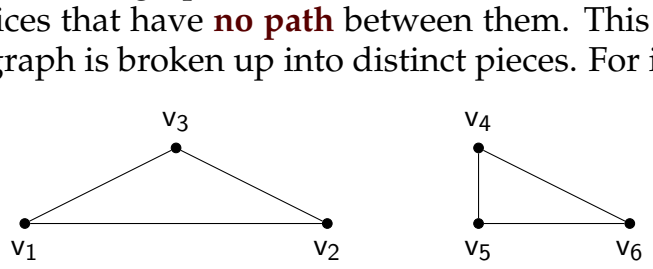
25.1.5 Connectedness

A graph is **connected** if there is a path between any two distinct vertices in the graph. Consider this graph again:



Between every pair of distinct vertices, there is a path. Pick any two distinct vertices, and you will be able to find a path between them.

By contrast, a graph is **disconnected** if there are at least two vertices that have **no path** between them. This happens when a graph is broken up into distinct pieces. For instance:

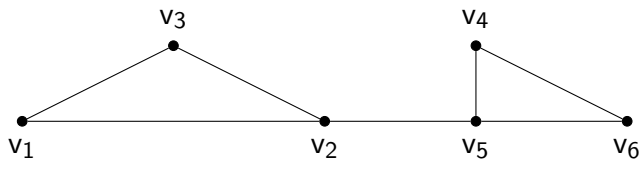


Remark 25.4. Notice that we can talk about how many "pieces" a graph is broken up into. In the first graph, there is just one "piece" (it is not broken up into pieces). In this second graph, there are two "pieces."

This graph is in two pieces: a left piece, comprised of vertices v_1, v_2, v_3, and the edges between them; and a right piece, comprised of vertices v_4, v_5, v_6, and the edges between them.

Notice that there is no way to travel from one piece to another, because there is no edge connecting the left piece to the right piece. Consequently, there are vertices in this graph that have no path between them. Just pick any vertex from the left piece and any vertex from the right piece. You will see that there is no path between them.

Suppose there were a single vertex to connect these two pieces. For instance, suppose the graph had this extra edge between v_2 and v_5:



The edge between v_2 and v_5 is called a **bridge**, because it bridges the gap between the two pieces. With the bridge in place, this graph is a **connected** graph, because now there is a way to travel across the gap. So no matter which two vertices we pick, we will be able to find a path between them.

Remark 25.5. To say that there is **no path** between two vertices exactly characterizes what it means for a graph to be disconnected. If you can't travel between two points, then there is a "gap" between them, i.e., a place where there are no edges. So you must have two pieces. Likewise, to say that **there is** a path between any two vertices exactly characterizes what it means for a graph to be connected. If you can travel between any two points, then there are no "gaps" between two pieces that you cannot cross.

25.2 Null and Complete Graphs

TWO KINDS OF GRAPHS crop up so often that graph theorists have given them names: *null* graphs and *complete* graphs.

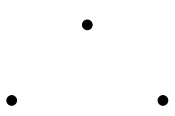
25.2.1 Null Graphs

A **null graph** of order n is a graph with n vertices and no edges. In other words, a null graph is a graph whose vertices have the most *minimal* degree (i.e., a degree of zero).

Here is the null graph of order 1 (we will call it $\mathbf{N}_1$):

It is just a single point, with no edges. Here is the null graph of order 2 (we will call it $\mathbf{N}_2$):

It is two points, with no edges. Here is the null graph of order 3 ($\mathbf{N}_3$):



It is three points, with no edges. We could go on, but you get the picture. We can keep increasing n by one to get $\mathbf{N}_4$, then $\mathbf{N}_5$, and so on.

Terminology. A **null graph** of order n has n vertices with **no edges**. In other words, each vertex in the graph has the absolute minimal degree (i.e., each vertex has a degree of zero). We denote a null graph of order n like this: $\mathbf{N}_n$, e.g., $\mathbf{N}_3$, $\mathbf{N}_4$, and so on.

25.2.2 Complete Graphs

A **complete** graph of order n is a graph with n vertices, each of which are connected every other vertex. Basically, a complete graph is a graph that has as many connections as possible between all the points. Each vertex has the *maximal* degree (the maximum amount of edges).

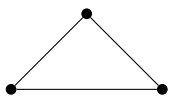
Here is the complete graph of order 1 (we will call it $\mathbf{K}_1$):

There is just one point here, so there are no edges. Hence, $\mathbf{K}_1$ is the same graph as $\mathbf{N}_1$.

Here is the complete graph of order 2 (we will call it $\mathbf{K}_2$):

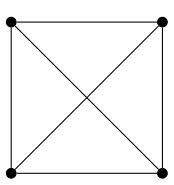
It has two points, with a single edge, because that's the most connections there can be between two vertices. Here is the complete graph of order 3 ($\mathbf{K}_3$):

Terminology. A **complete graph** of order n has n vertices with **maximal edges**. In other words, each vertex in the graph has the maximum degree possible (i.e., each vertex has an edge to every other vertex). We denote a complete graph of order n like this: $\mathbf{K}_n$, e.g., $\mathbf{K}_3$, $\mathbf{K}_4$, and so on.

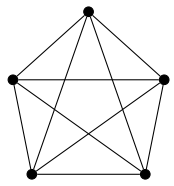


This one has three vertices, and three edges, because that is the most connections there can be between three points. Each point is connected to every other point.

Here are the complete graphs of order 4 and 5 (we will call them $\mathbf{K}_4$ and $\mathbf{K}_5$):



$\mathbf{K}_4$



$\mathbf{K}_5$

Remark 25.6. What is the **degree** of a complete graph of order n? Notice that it is always $n-1$, i.e., one less than its **order**. Why is that? Because: by definition a complete graph connects each vertex to every other vertex. If there are n vertices, then each vertex will be connected to every vertex except itself (so all n of them except for one). Hence, there will always be $n-1$ edges coming out of each vertex in a complete graph of order n.

25.3 Summary

IN THIS CHAPTER, we learned about some of the different properties of graphs.

- The **order** of a graph is the number of vertices it has. The **degree** of a vertex is the number of edges it has.
- A **path** (synonymously, a **walk**) in a graph is a trail from one vertex to another vertex in the graph, going from vertex to vertex by traveling across edges. A **cycle** (synonymously, a **circuit**) is a path that starts and ends at the same point.
- A graph is **connected** if there is a path between any two vertices (which means the graph is in one "piece"). A graph is **disconnected** if there are some vertices that there

is no path between (this happens when the graph is in two or more "pieces").

- Null graphs of order n (e.g., $\mathbf{N}_1$, $\mathbf{N}_2$, and so on) are graphs that contain n vertices and no edges. Complete graphs of order n (e.g., $\mathbf{K}_1$, $\mathbf{K}_2$, and so on) are graphs that contain n vertices and as many connections as possible (i.e., every vertex is connected to every other vertex).

[26]

GRAPH ISOMORPHISMS

IN CHAPTER 17, we talked about isomorphic sets, and we said that two sets are **isomorphic** if they have the same shape. The same concept exists for graphs: two graphs are **isomorphic** if they have the same shape too. In this chapter, we will talk about **maps** (functions) between graphs, and **graph isomorphisms**.

Ponder. What do you think it means to say that two graphs have the same “shape”?

26.1 Mapping Vertices

TO CONSTRUCT a **map** from one graph to another graph, all we need to do is map the vertices of one to the vertices of the other.

Terminology. A **map** from one graph to another is a function that maps the vertices of the one to vertices of the other.

Example 26.1. Suppose we have two graphs, called **G** and **H**:

$$\mathbf{G} = (V, E) \qquad \mathbf{H} = (W, D)$$

Suppose also that for $\mathbf{G}$, the vertices V and the edges E are defined like this:

$$V = \{v_1, v_2, v_3\}$$
$$E = \{(v_1, v_2), (v_2, v_3)\}$$

And suppose that for $\mathbf{H}$, the vertices W and edges D are defined like this:

$$W = \{v_5, v_6, v_7, v_8\}$$
$$D = \{(v_5, v_7), (v_6, v_7), (v_7, v_8)\}$$

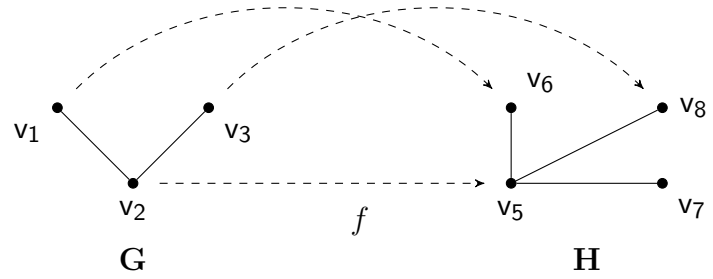
Let's draw these two graphs:

$\mathbf{G}$ $\mathbf{H}$

Now let's create a mapping (a **function**) from $\mathbf{G}$ to $\mathbf{H}$. Let's call our function f. To create this function, we need to map the vertices of $\mathbf{G}$ to the vertices of $\mathbf{H}$. So, all we need to do is go through each vertex of $\mathbf{G}$, and map it to some vertex of $\mathbf{H}$. For example, let's define f like this:

Remark 26.1. Since we are talking about two graphs here, we use different names for the vertices and edges, so that we can tell them apart. For $\mathbf{G}$ we call them V and E, but for $\mathbf{H}$ we call them W and D.

Remark 26.2. A good exercise is to try and draw these two graphs yourself, using just the sets of vertices and edges listed above. See if you can make your drawings look different from the drawings here.



In other words, here is the mapping:

$$f(v_1) = v_6 \qquad f(v_2) = v_5 \qquad f(v_3) = v_8$$

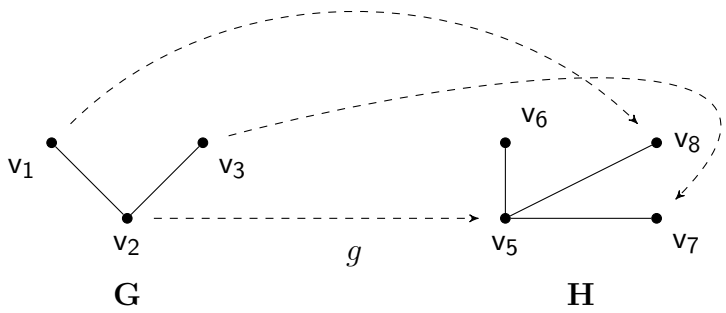
Since this function f maps the graph $\mathbf{G}$ to the graph H, let's denote the **signature** of this function like so:

$$f : \mathbf{G} \to \mathbf{H}$$

Read that like this: "the function f maps the graph $\mathbf{G}$ to the graph $\mathbf{H}$," or even, "f maps $\mathbf{G}$ to $\mathbf{H}$."

Remark 26.3. Recall from Chapter 14 that a **function** from one set to another set is a mapping that maps each element from the first set to one element from the second set.

Example 26.2. Let's construct a different mapping. Let's call this one $g : \mathbf{G} \to \mathbf{H}$, and define it like this:



Remark 26.4. What is this function as a list of pairings? It is this: $\{(v_1, v_8), (v_2, v_5), (v_3, v_7)\}$.

Here is the mapping:

$$g(v_1) = v_8 \qquad g(v_2) = v_5 \qquad g(v_3) = v_7$$

This function g differs from f because it maps two of the vertices of $\mathbf{G}$ to different vertices in $\mathbf{H}$.

26.2 Preserving Structure

NOT ALL MAPS from $\mathbf{G}$ to $\mathbf{H}$ are equal. Some maps **preserve the structure** of $\mathbf{G}$ in $\mathbf{H}$. What this means is that connected vertices from the first graph are mapped to similarly connected vertices in the second graph.

Terminology. A function from one graph to another **preserves the structure** if it maps connected elements in the first set to connected elements in the second set. It preserves the connections.

Let's be a little more exact. Consider a function $f : \mathbf{G} \to \mathbf{H}$. Take any two vertices x and y from $\mathbf{G}$. The function f maps x to some vertex in $\mathbf{H}$ (which we denote as $f(x)$), and it also maps y to a vertex in $\mathbf{H}$ (which we denote as $f(y)$).

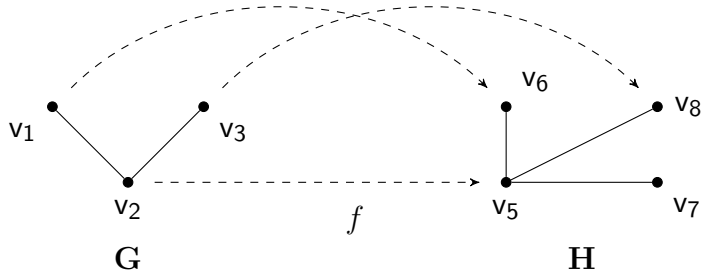
Remark 26.5. Recall from Chapter 14 that if we have a function f from A to B, then we denote the element in B that f maps x to like this: "$f(x)$," which we can read as "the element that f maps x to," or for short, "f of x."

To say that f preserves the structure of $\mathbf{G}$ in $\mathbf{H}$ is to say that whenever x and y are connected in $\mathbf{G}$, then f maps x and y to points $f(x)$ and $f(y)$ that are also connected in $\mathbf{H}$.

In other words, if the points in $\mathbf{G}$ are connected, then the points f maps them to in $\mathbf{H}$ are connected too. Let's put this down as a definition.

Definition 26.1 (Graph structure-preserving functions)**.** For any graphs $\mathbf{G}$, $\mathbf{H}$, and for any function $f : \mathbf{G} \to \mathbf{H}$, we will say that f is a **structure-preserving function** if whenever two vertices x and y are connected in $\mathbf{G}$, $f(x)$ and $f(y)$ are connected in $\mathbf{H}$.

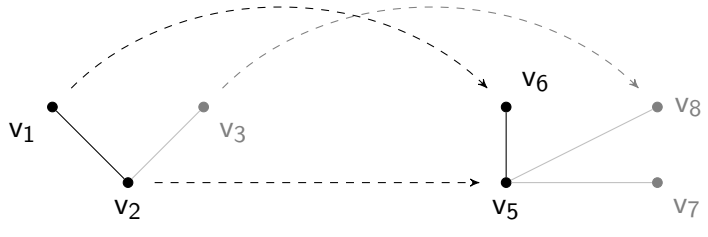
Example 26.3. Consider $f : \mathbf{G} \to \mathbf{H}$ again:



Does f preserve the structure of **G**? To determine if this is so, we need to check every connection in **G**, and check whether f preserves it in **H**. There are two connections **G** we need to check: the connection of v_1 and v_2, and the connection of v_2 and v_3.

First let's check v_1 and v_2. Does f map v_1 and v_2 to points in **H** that are also connected? Yes, it does: f maps v_1 to v_6 (i.e., $f(v_1) = v_6$), and it maps v_2 to v_5 (i.e., $f(v_2) = v_5$), and v_5 and v_6 are connected in **H**, just as v_1 and v_2 are in **G**.

Here is the picture with just this part of the mapping highlighted:

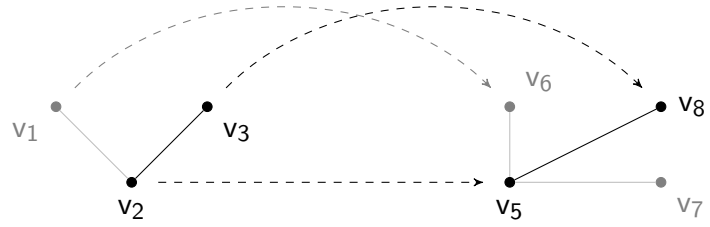


Now let's check the connection between v_2 and v_3. Does f preserve this connection too? Yes, it does. Here is the picture with just that part of the mapping highlighted:

Remark 26.6. To **prove** that a map from one graph to another **preserves the structure**, we need to check **each** connection in the first graph, and confirm that it is preserved by the mapping.

Remark 26.7. In **G**, there are two connections: v_1-v_2 and v_2-v_3. We need to check that each one is preserved by f.

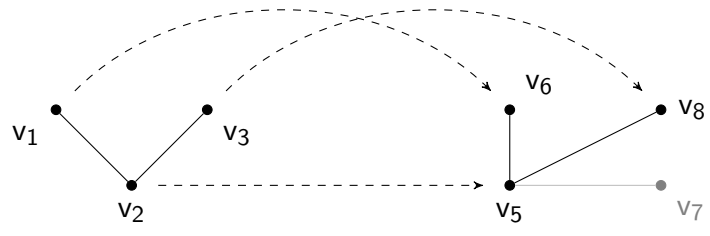
Remark 26.8. In the picture, we can see the mapping of the first connection from **G**. We can see that v_1-v_2 is mapped to v_5-v_6. If v_1 and v_2 were mapped to points in **H** that weren't connected, then their connection would not *not* be preserved. But here we can see that v_1 and v_2 are mapped to points in **H** that are connected, so their connection is preserved.



Remark 26.9. In the picture, we can see the mapping of the second connection from **G**. We can see that v_2-v_3 is mapped to v_5-v_8, so the connection here is preserved too.

We can see here that f preserves the connection between v_2 and v_3, because it maps those points to v_5 and v_8, which are connected in **H**.

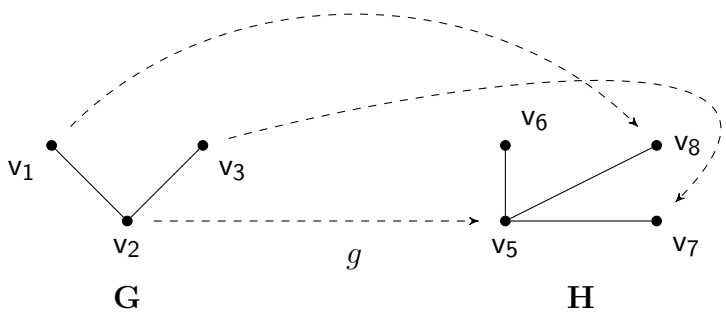
We have now checked all the connections in **G**, and confirmed that f preserves each one in **H**. Here is a picture of all of the connections, as they are mapped:



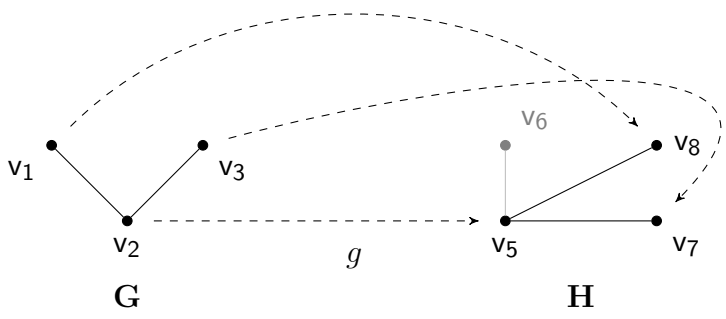
Remark 26.10. If you squint, you can see that the highlighted part of **H** has the same structure as **G**: both have a "V"-shape. Notice that not every part of **H** is covered by the mapping. In particular, v_5-v_7 makes for a rogue limb that is not part of the mapping. But that is okay. A structure-preserving map need not map **G** to *every* part of **H** (in other words, it does not need to be **surjective**). All it needs to do is preserve the structure of **G** in the part of **H** that it does map to. We might say that in this case, our function f **embeds** **G** in **H**.

We can see clearly in this picture that f preserves the structure of **G**. It takes a "V"-shaped graph **G**, and it maps that to a "V"-shaped portion of **H**, preserving the structure. Hence, we may conclude that f is a **structure-preserving function**.

Example 26.4. Consider $g : \mathbf{G} \to \mathbf{H}$ again:



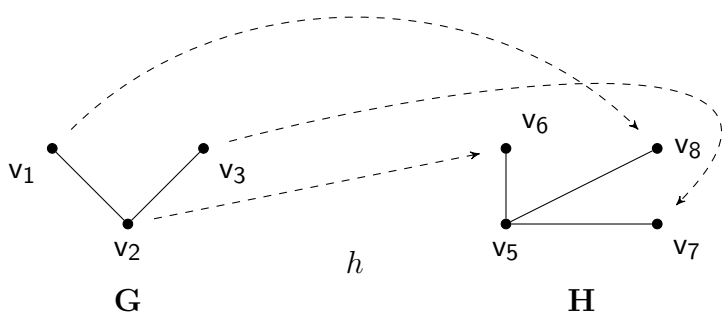
Is g a structure-preserving map? Here is the picture, with just the relevant part of the mapping highlighted:



We can see from this that g does indeed preserve the structure of **G** in **H**. This too maps a "V"-shaped graph **G** to a "V"-shaped portion of **H**. We can confirm this manually, by checking each connection from **G**, and confirming that the connection is preserved in **H**.

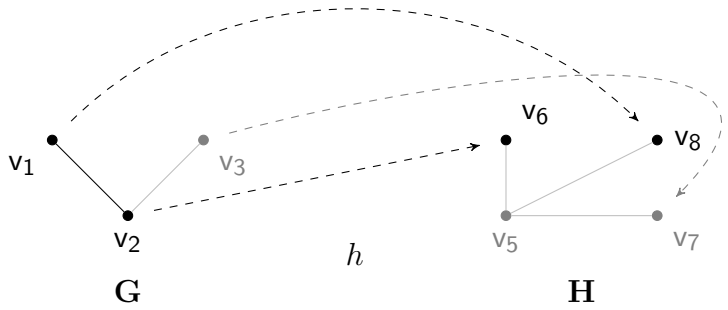
Remark 26.11. Here too we can see that g **embeds** **G** in **H**. It takes the "V"-shape of **G**, and it maps it to a "V"-shaped portion of **H**. Notice that **H** is a bigger graph than **G**, and it has two "V"-shapes in it. That is why it is possible to construct two different structure-preserving maps from **G** to **H**. It is because there are two ways to embed the "V"-shape of **G** into **H**.

Example 26.5. Consider this function $h : \mathbf{G} \to \mathbf{H}$:



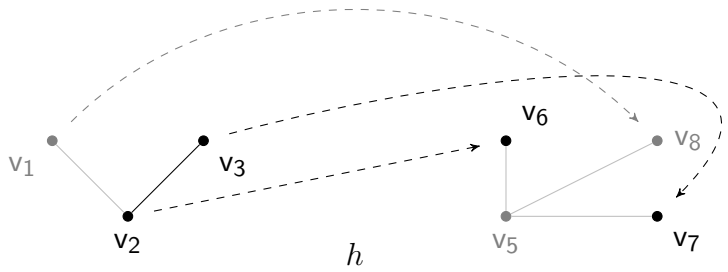
Is h a structure-preserving map? In order to determine this, we need to check that it preserves every connection from **G**. Let's check how it maps v_1-v_2 first:

Remark 26.12. What are the pairings of this function? It is this: $\{(v_1, v_8), (v_3, v_7), (v_2, v_6)\}$.



Does h preserve the connection between v_1 and v_2? No, it does not. It maps v_1 and v_2 to v_6 and v_8, which are *not* connected in **H**. What about v_2-v_3?

Remark 26.13. To **prove** that a map is **not** a structure-preserving map, all we need to do is find one connection that it does not preserve. So, since we have now shown that v_1-v_2 is not preserved, we have succeeded in showing that h is not a structure-preserving map. For the sake of illustration though, we will also look at v_2-v_3, because h doesn't preserve that connection either.



Does h preserve the connection between v_2 and v_3? No, it does not. It maps v_2 and v_3 to v_6 and v_8, which are also *not* connected in $\mathbf{H}$.

So h fails to preserve the connections from $\mathbf{G}$, and hence, h is **not** a structure-preserving map.

26.3 Graph Isomorphisms

WHAT IS A GRAPH ISOMORPHISM? A **graph isomorphism** from $\mathbf{G}$ to $\mathbf{H}$ is a structure-preserving map from $\mathbf{G}$ to $\mathbf{H}$, which is bijective and reversible. That is to say, it puts the vertices of $\mathbf{G}$ and $\mathbf{H}$ into a one-to-one correspondence, such that the structure of $\mathbf{G}$ is exactly mirrored in $\mathbf{H}$, and vice versa. Let's write this down as a definition:

Terminology. A **graph isomorphism** is a reversible, bijective, structure-preserving map from one graph to another.

Definition 26.2 (Graph isomorphism)**.** For any graphs $\mathbf{G}$, $\mathbf{H}$, and for any function $f : \mathbf{G} \to \mathbf{H}$, we will say that f is a **graph isomorphism** if f is a reversible, bijective, structure-preserving map from $\mathbf{G}$ to $\mathbf{H}$.

Remark 26.14. Think about what two graphs must be like in order for us to be able to build a reversible, bijective, structure-preserving map between them. First, they must have the **same number of vertices**. If one had more, we couldn't build a reversible bijection between them. Second, they must have the **same connections**. If they didn't, then we couldn't build a structure preserving map between them. So basically, the only case where we can build an isomorphism between two graphs is when those two graphs are exactly alike, except for having different vertex names.

It is not always possible to construct an isomorphism between two graphs. It it only possible when the two graphs have the same number of vertices, and when they mirror each other's structure. Basically, it is only possible if the two graphs differ only in the *names* of their vertices.

If we can construct an isomorphism between a graph $\mathbf{G}$ and a graph $\mathbf{H}$, then we say the two graphs are **isomorphic**, and we write that like this:

$$\mathbf{G} \cong \mathbf{H}$$

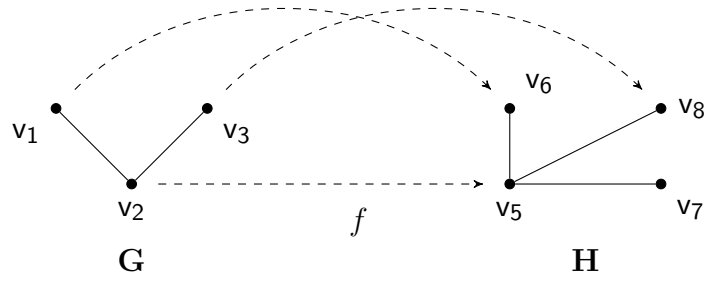
Terminology. If we can build a graph isomorphism between two graphs $\mathbf{G}$ and $\mathbf{H}$, then those two graphs are **isomorphic**. We denote that like this: $\mathbf{G} \cong \mathbf{H}$.

Read that aloud like so: "the graph $\mathbf{G}$ is isomorphic to the graph $\mathbf{H}$," or "the graphs $\mathbf{G}$ and $\mathbf{H}$ are isomorphic." Let's put this down as a definition too:

Definition 26.3 (Isomorphic graphs)**.** For any graphs $\mathbf{G}$ and $\mathbf{H}$, we will say that $\mathbf{G}$ and $\mathbf{H}$ are **isomorphic** if there is a graph isomorphism $f : \mathbf{G} \to \mathbf{H}$ between them. To denote that they are isomorphic, we will write this: $\mathbf{G} \cong \mathbf{H}$.

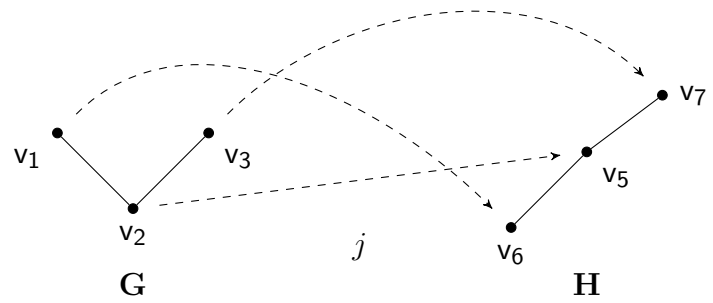
Remark 26.15. Although f preserves the structure of $\mathbf{G}$ in $\mathbf{H}$, it does not completely cover $\mathbf{H}$. There is more to $\mathbf{H}$ than there is to $\mathbf{G}$. In particular, there is a rogue limb v_5-v_7 in $\mathbf{H}$ that $\mathbf{G}$ has no twin counterpart for. Hence, although we can map the structure of $\mathbf{G}$ into $\mathbf{H}$, we cannot **reverse** it. We cannot embed the structure of $\mathbf{H}$ back into $\mathbf{G}$. We can see that this is so simply from the fact that the two graphs have a different number of vertices (their **order** is different). From that one fact alone, we can deduce that there **is no isomorphism** between these two graphs (and hence that they are not isomorphic).

Example 26.6. Consider $f : \mathbf{G} \to \mathbf{H}$ again:



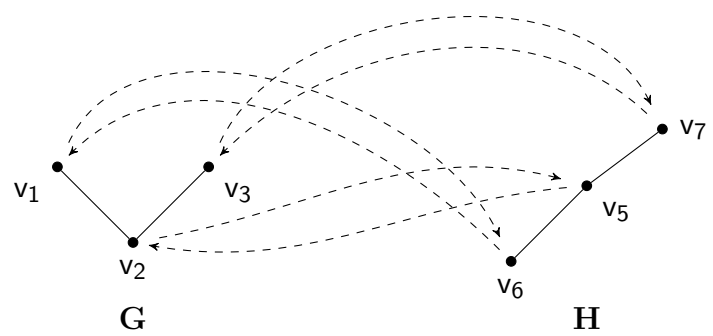
Are $\mathbf{G}$ and $\mathbf{H}$ isomorphic? The answer is no. The reason is that f is not bijective and reversible. There is an extra limb in $\mathbf{H}$ that is not in $\mathbf{G}$, and so the two graphs clearly don't have *exactly* the same structure. We can embed $\mathbf{G}$ in $\mathbf{H}$, but we cannot go the other way around!

Example 26.7. Consider this mapping $j : \mathbf{G} \to \mathbf{H}$:



This is a graph **isomorphism**. It is *structure-preserving* (check that it is so for yourself), and it is **bijective**. It is also reversible, which is clear because we can map the end-points of j right back to their start-points:

Remark 26.16. Notice that these two graphs have exactly the same number of vertices (they have the same **order**), and they have the same **connections**. That is why it is possible to construct an isomorphism between them.



This shows that these two graphs have the same shape (and in fact you can see it if you bend back v_1 to make **G** look more like a straight line). The only real difference between them is that their vertices have different names:

- The left vertex is named v_1 in **G**, and v_6 in **H**.
- The middle vertex is v_2 **G**, and v_5 in **H**.
- The right vertex is v_3 in **G**, and v_7 in **H**.

Since j is an isomorphism, we may conclude that **G** and **H** are indeed **isomorphic**. That is, $\mathbf{G} \cong \mathbf{H}$.

Remark 26.17. Recall from Chapter 17 that a function f is **reversible** if we can build a second function (which we denote as f^{-1}) that maps the end-points of f right back to where they started.

26.4 Summary

IN THIS CHAPTER, we learned about **graph isomorphisms**.

- A **map** from a graph $\mathbf{G}$ to a graph $\mathbf{H}$ is a function that maps the vertices of $\mathbf{G}$ to the vertices of $\mathbf{H}$. To denote the signature of a map f from a graph $\mathbf{G}$ to a graph $\mathbf{H}$, we write this: $f : \mathbf{G} \to \mathbf{H}$.
- A map $f : \mathbf{G} \to \mathbf{H}$ is **structure-preserving** if it preserves the connections of $\mathbf{G}$ in $\mathbf{H}$. More exactly, if whenever two vertices x and y are connected in $\mathbf{G}$, $f(x)$ and $f(y)$ in $\mathbf{H}$ are also connected.
- A structure-preserving map $f : \mathbf{G} \to \mathbf{H}$ is a **graph isomorphism** if it is bijective and reversible. If we can construct such an isomorphism from $\mathbf{G}$ to $\mathbf{H}$, then that shows us taht $\mathbf{G}$ and $\mathbf{H}$ are **isomorphic**. They are basically mirror images of each other, and they differ only in the names of their vertices.

[27]

OTHER KINDS OF GRAPHS

IN CHAPTER 24 WE DEFINED A GRAPH as a set of objects with some connections. If we add some restrictions on the kinds of connections that we allow, then we can distinguish some further kinds of graphs. In particular, we can distinguish **simple graphs**, **multigraphs**, **pseudographs**, and **directed graphs**. Usually, we deal with simple graphs, but it's worth knowing about the other types.

27.1 Simple Graphs

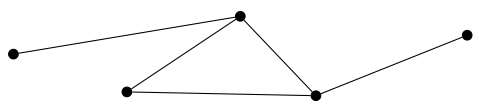
A **simple graph** is a graph with the following properties:

- There can be at most only **one edge** between vertices. In a drawing, that means there can be at most only one line

between two points. There cannot be two lines (edges) going between points.

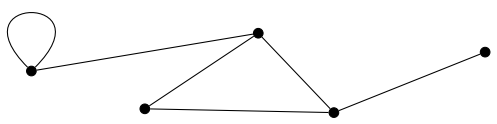
- **No self-loops** are allowed.

Here is an example of a simple graph:



We can see that there is at most one edge between any two vertices, and there are no self-loops.

Here is an example of a graph that is not a simple graph:



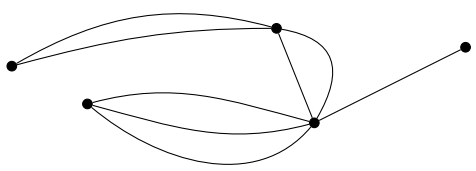
This fails to be a simple graph because it has a self-loop.

27.2 Multigraphs

A **multigraph** is a graph with the following properties:

- There can be **many edges** between vertices. In a drawing, that means there can be more than one line between two points.
- In some math books, multigraphs are not allowed to have self-loops. In other math books, self-loops are allowed. When reading about graphs, it is always best to check how the authors define multigraphs.

Here is an example of a multigraph:



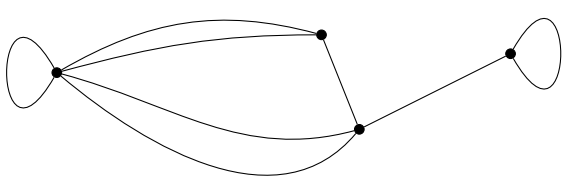
We can see that this graph has more than one edge between some of the points, so it is multigraph.

27.3 Pseudographs

A **pseudograph** is a graph with the following properties:

- **Self-loops** are allowed.
- In some math books, **many edges** between vertices are allowed, like multigraphs. But in other math books, pseudographs do not allow multiple edges between the same points. When reading about graphs, it is always best to check how the authors define pseudographs.

Here is an example of a pseudograph:



Unlike a simple graph, we can see that this graph has self-loops, so it is a pseudograph.

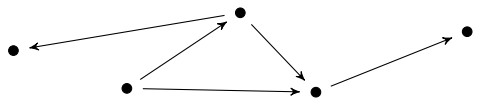
27.4 Directed Graphs

Usually, when we talk about graphs, the edges represent a two-way connection. If a is connected to b, then b is connected to a, by the same edge.

However, there are cases where we want the connections to go only one-way. (For instance, maybe we are trying to model all the one-way streets in a city.) When the direction of the edges matter, we draw the edges as arrows, and we call the graph a **directed graph**. So a directed graph is a graph with this property:

- Each edge goes one-way: one vertex is the **source**, and the other is the **destination**, of the edge.

Here is an example of a directed graph:



Directed graphs can be simple, multi, or pseudo. If they have at most one edge between points and no self-loops, then it's a directed simple graph. If there are multiple edges, it is a directed multigraph. If it allows self-loops as well, then it is a directed pseudograph.

27.5 Summary

IN THIS CHAPTER, we learned about a few different types of graphs.

- **Simple graphs** are graphs with no self-loops and at most one connection between any two vertices.

- **Multigraphs** are graphs that allow more than one connection between any two vertices. Some math books allow multigraphs to have self-loops, but other math books do not. It is always best to check how the authors of whatever book you are reading actually define multigraphs.
- **Pseudographs** are graphs that allow self-loops. Some math books allow pseudographs to have multiple edges between the same vertices, but other math books do not. It is always best to check how the authors of whatever book you are reading actually define pseudographs.
- **Directed graphs** are graphs where the direction of the connection matters. We usually draw the connections with arrows instead of lines, so we can indicate which direction the connection goes.

[28]

FURTHER READING

To pursue graph theory further, the following list may offer some helpful starting points.

- Burger and Starbird (2010, ch. 5) offers an introductory level discussion of a few different topics in graph theory. This is an excellent place to start reading first.

- Trudeau (1993(@) provides an excellent introduction to graph theory, and to higher mathematics in general. The pace is slow, and the proofs are fully explained. It offers a good entry point to graph theory for the beginner, and going through this text slowly can really pay off.

- Chartrand (1977) provides an introductory discussion of a number of topics in graph theory.

- Jongsma (2019, ch. 8) offers a rigorous introduction to the basic theorems and proofs behind a couple of topics in graph theory.
- Flegg (1974, chs. 6–8) provides a more conceptual discussion of some of the classic concepts in graph theory.
- Benjamin et al. (2015) presents a variety of classic topics from graph theory. The authors provide a good amount of historical detail, as well as decent discussion of the classic theorems and proofs.
- Hartsfield and Ringel (1990) is another thorough introduction to classic topics in graph theory.
- Fournier (2009) provides a nice discussion of many applications of graph theory. Its discussion of the math itself is often fairly brief, so it is useful mainly for its discussion of applications.
- Gould (2012) offers a very thorough discussion of the classic topics of graph theory. One of the great features of this book is that it discusses the algorithms behind many of these topics, and so it is very useful for computer scientists.

Part [6]

NUMBERS

What exactly are **numbers**? How do we define a number without secretly relying on the concept of number? And how do we count, anyway? What about negative numbers, and fractions, or numbers with decimals points in them?

In Part 6, we discuss four different **sets of numbers**. We look at the most basic set of numbers, which we use to *count*, called the **natural numbers**. Then we look at the **integers**, which includes negative numbers so we can keep track of adding and taking away. Next, we look at the **rational numbers**, which are the set of fractions. Finally, we look at the **real numbers**, which is the set of every point on a real number line.

[29]

THE NATURAL NUMBERS

WHAT EXACTLY IS a **number**? How do we even define numbers? If you think about it, it's quite difficult to come up with a definition of number that doesn't rely on the idea of number.

But mathematicians are logical folk, and they really like to not have circular definitions. They like to try and define things in terms of simpler concepts. So, the task is: can we define numbers in simpler terms, without secretly relying on the concept of number?

Since the beginning of civilization, humans have been working at understanding numbers. It turns out there is actually more than one type of number, which we will talk about in this and the next few chapters. In this chapter, we will start with the simplest type of numbers, namely the set of numbers we use to count.

Ponder. How would you define the concept of “a number”? Can you think up a definition for “number” that does not involve using the concept of number? A true definition will not be circular.

29.1 Counting

WE ALL KNOW THE COUNTING NUMBERS. It is easy for us to write them down. For instance, we can start out our listing like this:

$$0, 1, 2, 3, 4, \ldots$$

Notice, however, that these are really just **symbols** that we use for counting. They are nothing more than **squiggly marks** that we draw on the page. What really *are* they? What do they mean? Let's look at counting more carefully, to see if we can understand these numbers better.

Ponder. It's a good exercise to stop before reading on, pretend that you don't know the symbols "0," "1," "2," and so on, and then think about how you might count objects and keep records of your totals. How might you do it?

What exactly happens when we count things? Suppose we are looking at a bunch of apples lying on a table in front of us, and we want to count them. Let's pretend that we don't know the familiar symbols "0," "1," "2," and so on, so we can't count our apples using the counting numbers we all learned in school. But let's suppose that we do have a pencil and a blank piece of paper, that we can use to record our progress. How might we count the apples?

Well, before we start counting the apples, we start with a tally of nothing. To symbolize this, let's just make a special mark "$\oslash$" on our paper. Hence, our paper looks like this:

Total tally: $\oslash$

Remark 29.1. To symbolize nothing, we could just write no marks on the page at all and leave our paper blank. But it's more explicit to actually write down a symbol that says "The tally so far comes up to nothing."

That symbol just means that we have counted nothing. So we can read it like this: "Total tally: nothing."

Next, we point to the first apple, and we count it. We can't use number symbols, so maybe we just write a scratch "$\rceil$" on our piece of paper, to indicate that we have counted one more than we had before. So, our paper now looks like this:

Total tally: ⊘-|

We can read that like this: "Total tally: one more than nothing."

Next, we point to the second apple, and we count it. Again, we can't use number symbols, so we write down another scratch "|" to indicate that we have counted yet one more than before. Our paper now looks like this:

Total tally: ⊘-|-|

We can read that like this: "Total tally: one more than one more than nothing." To count the third apple, we add another tally mark:

Total tally: ⊘-|-|-|

Which we can read as "Total tally: one more than one more than one more than nothing."

We could go on like this, counting one apple after another by adding a new tally for each apple we come to.

Isn't this exactly what happens when we count things? We go through them, one by one, building up a tally. At each point, we say we have "one more" than before.

So really, we have just two ideas here. We have a **starting point** (which we symbolized as "⊘"), and then we have the idea of "one more than before," i.e., the **successor** (which we symbolized with a tally mark "|").

Notice that when we count our apples like this, we never use the symbols "0," "1,", "2," and so on. In fact, we never even use those *concepts*. Of course, we do say that each tally mark represents "*one* more than we had before." But we're not really using the numerical concept of "one" here. We don't mean "1" when we say that. We mean something much closer to "another of them."

Remark 29.2. Each tally mark indicates that we have counted one more than we had before. Each tally mark means something like this: "another one."

Remark 29.3. In fact, we could go on counting apples like this **infinitely**, if we lived forever and had an infinite supply of apples.

Terminology. At a basic level, when we count, there are two things we use. First, we use a **starting point**, then we repeatedly use the idea of the **successor**, i.e., "one more than before." None of this requires that we understand or use the concepts of "0," "1," "2," and so on.

29.2 Definition and Notation

NOW THAT WE KNOW WHAT COUNTING IS, let's use it to define a set of counting numbers. Let's call this set the **natural numbers**, and let's symbolize it like this:

$$\mathbb{N}$$

Read that out loud like this: "the set of natural numbers" (or, if you like, you can just call it the letter "N").

Of course, there are a lot of counting numbers, and it would be tedious to list out every one in a set roster. So, let's instead describe how to *build* this set.

First, let's say that $\mathbb{N}$ includes a special object which represents the starting point for any counting. Let's call this **nought**, and let's denote it with the symbol "$\oslash$" from before.

It might be tempting to think of this symbol as representing "nothing" or "zero." But we don't necessarily need to think of it as representing "nothing." We can just think of it as "the *starting point* for counting."

At this point, our set of natural numbers $\mathbb{N}$ includes one object, namely nought. Let's write out what we have so far:

$$\mathbb{N} = \{\oslash\}$$

We can read that out loud like this: "the set of natural numbers includes nought."

Next, let's say that we can add a new object to $\mathbb{N}$ by taking any object already in $\mathbb{N}$, and appending to it a tally mark. Let's interpret the tally mark as "the (next) one after ...," or the **successor** of it. Let's write this out as a rule, like this:

$$\text{if } \text{“}n\text{”} \in \mathbb{N}, \text{ then } \text{“}n\text{-}\rceil\text{”} \in \mathbb{N}$$

Terminology. The **natural numbers** is the set of counting numbers. We denote it like this: $\mathbb{N}$.

Remark 29.4. There is **no end** to the counting numbers. For any number we count up to, there's always one more! After all, we can always add another tally mark. So it is actually impossible to list out every counting number in a set roster.

Remark 29.5. Nought need not be thought of as nothing or zero. It is really just a **starting point** for counting. In fact, some mathematicians prefer to leave zero out of the set of natural numbers. For them, the starting point isn't zero at all, but rather one. So it really makes no difference. Nought is just the starting point.

Read this out loud like so: "if n is in the set of natural numbers, then the successor of n is in the set of natural numbers."

Let's use this rule to add some objects to our set $\mathbb{N}$. At this point, we only have one item in the set, namely $\oslash$. Our rule says that we can take $\oslash$, and append a tally to it, to get a new object "$\oslash$-$\rceil$" (which we can read as "the successor of nought"). So let's add that to our set. Now our set has two objects in it:

$$\mathbb{N} = \{\ \oslash,\ \oslash\text{-}\rceil\ \}$$

We can read that like this: "the set of natural numbers includes nought, and the successor of nought."

Let's use the rule again, to add another new object to the set. Our rule says we can take any n from $\mathbb{N}$, and append a tally mark to it, to get a new object that we can put back into $\mathbb{N}$. So let's take out "$\oslash$-$\rceil$," and add another tally to it to get "$\oslash$-$\rceil$-$\rceil$," i.e., "the successor of the successor of nought." Then when we put that back in $\mathbb{N}$, our set ends up with three objects in it:

$$\mathbb{N} = \{\ \oslash,\ \oslash\text{-}\rceil,\ \oslash\text{-}\rceil\text{-}\rceil\ \}$$

We can read that like this: "the set of natural numbers includes nought, the successor of nought, and successor of the successor of nought."

You can see how we could keep going like this forever. Obviously, that would be tedious, so let's just stipulate up front that the set $\mathbb{N}$ is the set that contains every possible object that can be built by adding tally after tally in this fashion. Hence, let's say that $\mathbb{N}$ looks like this:

$$\mathbb{N} = \{\ \oslash,\ \oslash\text{-}\rceil,\ \oslash\text{-}\rceil\text{-}\rceil,\ \oslash\text{-}\rceil\text{-}\rceil\text{-}\rceil,\ \ldots\ \}$$

At this point, we have actually defined the set of natural numbers. We have said that this set begins with a starting

Remark 29.6. Note that "n" is just a **placeholder** for any object already in $\mathbb{N}$. Hence, if "n" is actually "$\oslash$" (nought), then "n-$\rceil$" (the successor of n) is actually "$\oslash$-$\rceil$" (the successor of nought). If "n" is actually "$\oslash$-$\rceil$" (the successor of nought), then "n-$\rceil$" (the successor of n) is actually "$\oslash$-$\rceil$-$\rceil$" (the successor of the successor of nought).

Remark 29.7. We could never live long enough to finish the task of adding tally after tally to fill out this set. Even if we lived eternally, there would *still* not be enough time to do it. Why? Because there is no end to this set. No matter how big a number we get to, there is always a bigger one: we can always add another tally mark! So in point of fact, we have to imagine this set $\mathbb{N}$ as if it were **already built** in its entirety (by God perhaps), even though it goes on forever.

point (nought), and then it is built up by adding successors, over and over again, forever. Let's write this down as a formal definition.

Remark 29.8. Notice that our definition here **specifies** a set. And, notice that we do it with words, by describing how it is built up. This is a perfectly fine way to specify a set. What is important is that our description is unambiguous. It should be crystal clear to any reader, how to build up the set.

Definition 29.1 (Natural numbers)**.** We will say that the **natural numbers** is a set which we will denote as $\mathbb{N}$. We define this set in the following way:

(i) $\mathbb{N}$ contains a special object called **nought**, which we will denote as "$\oslash$."

(ii) For any n in $\mathbb{N}$, $\mathbb{N}$ contains another object called the **successor** of n, which we will denote as "n-⌉."

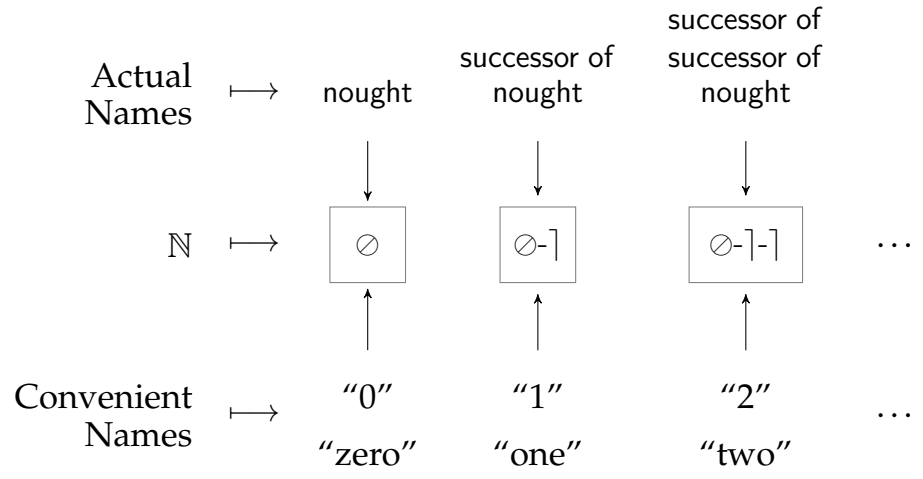
(iii) $\mathbb{N}$ contains no other objects besides those mentioned in (i) and (ii).

29.3 Familiar Names

Ponder. Which counting numbers do we use so often that it is useful for us to have a designated name for them? What about the first hundred natural numbers? What about in the thousands? Millions? Zillions?

IT IS QUITE TEDIOUS to speak about numbers in the above way. Suppose I ask you to bring me some ice cream sandwiches. You say, "how many do you want?" Imagine if I said, "the successor of the successor of the successor of nought." It is obvious that we should invent some names for some of these counting numbers that we use commonly.

That's where the numbers we all know and love come in. The symbols "0" (pronounced "zero"), "1" (pronounce "one"), "2" (pronounced "two"), and so on are just **names** for the natural numbers. They are short-hand abbreviations that we can use. So here's the picture:



You can see that, say, the squiggly mark "2" that we write down on paper, and pronounce as "two," is really just a convenient name for the successor of the successor of nought.

Remark 29.9. So we have unambiguous symbols like "⊘" or "⊘-⌉"; we have technical names like "nought" or "the successor of nought"; and we have convenient names and symbols like "zero" ("0") or "one" ("1").

29.4 More Standard Notation

ABOVE, WE USED TALLY MARKS to symbolize the natural numbers. But mathematicians don't usually use tally marks. They use a slightly different notation, which we will adopt from here on out. Let us call this the **standard notation**.

It goes like this. For nought, we use "0," and for the successor of n we write "$s(n)$." Using this notation, we can write the successor of nought like this:

$$s(0)$$

Terminology. To write the natural numbers in **standard notation**, we will write "0" for nought, and "$s(n)$" for the successor of n. The lowercase "s" in "$s(n)$" is a shorthand for "the successor."

And we can write the successor of the successor of nought like this:

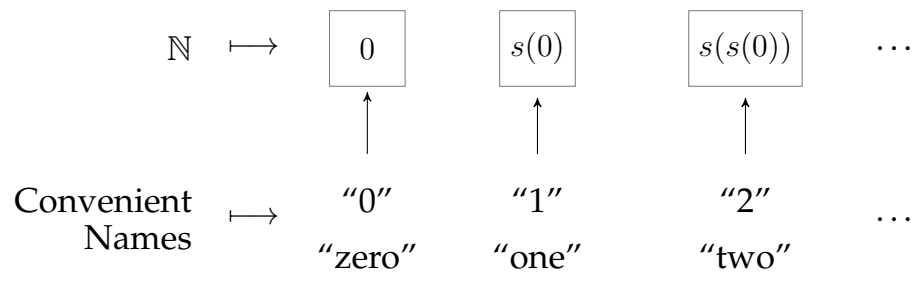
$$s(s(0))$$

Hence, in this more standard notation, we can denote the set of natural numbers $\mathbb{N}$ like this:

$$\mathbb{N} = \{0, s(0), s(s(0)), s(s(s(0))), \ldots\}$$

We still have convenient names like "one" and "two" for this set. All we did was change the notation for the natural numbers. Hence, we still have this picture:

Remark 29.10. Notice that you can easily determine which number a symbol represents by counting how many little "s" characters appear in it. For instance, "$s(s(0))$" represents two, because two "s" characters appear in it. And "0" represents zero because no "s" characters appear in it.



Sometimes mathematicians omit the parentheses so they don't have to write so much. In that case, $\mathbb{N}$ looks like this:

$$\mathbb{N} = \{0, s0, ss0, sss0, \ldots\}$$

Remark 29.11. Since mathematicians sometimes use different notation, whenever you read mathematical texts, it is always important to check what the *author* specifies as their notation.

There are various other, equivalent, ways that mathematicians sometimes use to write the natural numbers. Sometimes, they just use a tick mark for the successor. So they might write the natural numbers out like this:

$$\mathbb{N} = \{0, 0', 0'', 0''', \ldots\}$$

Another variant uses a little plus symbol:

$$\mathbb{N} = \{0, 0^+, 0^{++}, 0^{+++}, \ldots\}$$

However we choose to write it, it should be obvious that the basic idea is the same. The natural numbers are just counting numbers, and they are really nothing more than a sequence of successors after a starting point.

29.5 Summary

IN THIS CHAPTER, we learned about the counting numbers, which mathematicians call the **natural numbers**.

- If we think about counting, we see that counting is really nothing more than **starting** at a certain point, and then increasing a tally, **one at a time**.
- Formally then, we can define the natural numbers like this: the **natural numbers** are a set that contains a special symbol **nought**, and then for every n in the set, there is a **successor** of n that is also in the set. No further objects are in the set besides those just mentioned.
- We denote the set of natural numbers as $\mathbb{N}$. Using standard notation, we denote nought as "0," and we denote the successor of any n as "$s(n)$." Hence, the successor of nought is $s(0)$, the successor of the successor of nought is $s(s(0))$, and so on.
- The marks "0" (pronounced "zero"), "1" (pronounced "one"), "2" (pronounced "2"), and so on are just convenient **names** for the natural numbers. "0" is a name for nought, "1" is a name for the successor of nought, "2" is a name for the successor of the successor of nought, and so on.

[30]

INTEGERS

IN CHAPTER 29 we looked at the natural numbers, which we use for counting. In this chapter, we will look at another set of numbers, the **integers**. The integers include the natural numbers (the positive counting numbers), but they also include the negative numbers. So the integers is made up of both positive and negative counting numbers.

Ponder. Why do we need negative numbers? What things can you think of that you might need negative numbers for?

30.1 Adding and Multiplying

THE NATURAL NUMBERS INCLUDE no negative numbers. They include **positive numbers** only. Of course, this makes sense, since the natural numbers are the counting numbers. The process of counting is a process of *increasing* a tally, hence

Remark 30.1. The natural numbers include only positive numbers, no negative numbers.

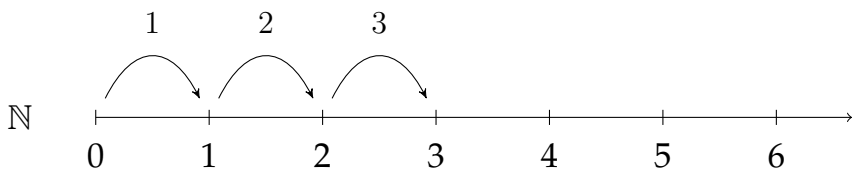
we need nothing more than a starting point, and then whole numbers going up from there.

Because of this, the natural numbers are particularly well suited for **adding** and **multiplying**. Consider adding m and n to get a sum p:

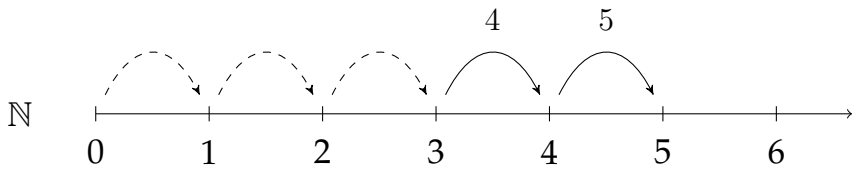
$$m + n = p, \text{ where } m, n, p \in \mathbb{N}$$

Terminology. To compute the answer to a problem with the form $m + n = p$, we add the numbers m and n together to get p. We call p the **sum** of m and n.

To compute p, we count m objects, and then we continue counting n more objects. For instance, to compute 3 + 2, I first count three objects:



Then I count two more:



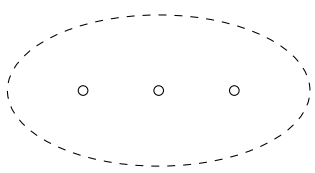
At the end of my counting, I end up at 5, so the sum of 3 and 2 is 5.

Multiplying is straightforward too. Consider multiplying m and n to get a product p:

$$m \cdot n = p$$

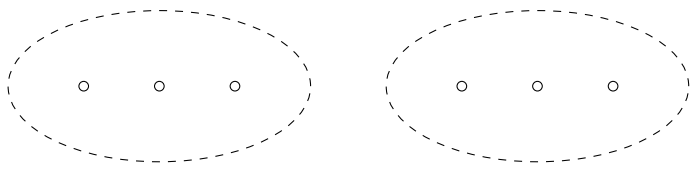
Terminology. To compute the answer to a problem with the form $m \cdot n = p$, we add together n groups of m objects, to get a total of p. We call p the **product** of m and n.

To compute this, we add together n groups each containing m objects. For instance, to compute $3 \cdot 2$, I put together a group of 3 objects:



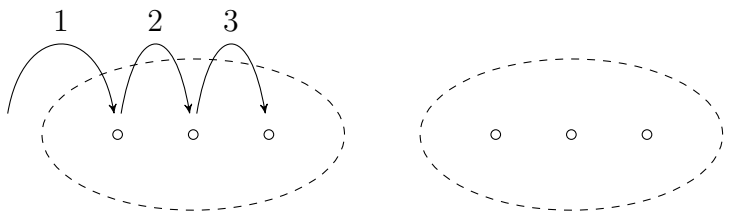
One group of 3

And then I put together another group of 3 objects:



One group of 3 Another group of 3

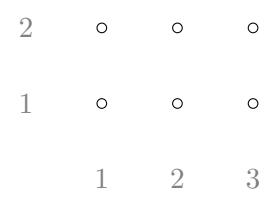
Now I have 2 groups of 3 (i.e., I have 2 bags, and each one contains 3 objects). Next, I just need to add them all up. So I count the first group:



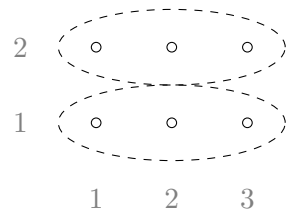
One group of 3 Another group of 3

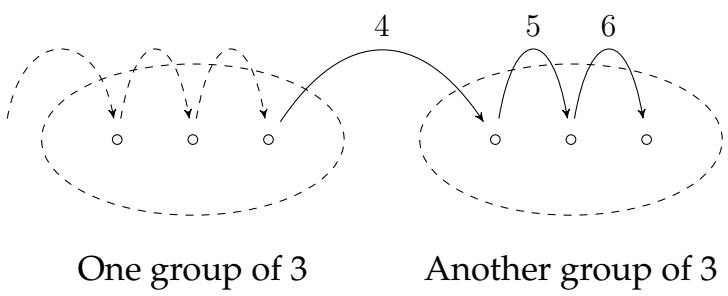
Which brings me up to 3, and then I continue counting into the second group:

Remark 30.2. Another way to visualize multiplying 2 and 3 is to make a 2×3 grid:



That also gives us two groups of three. They're just stacked vertically:





Ponder. Recall from Chapter 11 that the **product** of two **sets** can be represented by drawing a grid. Let $m = \{1, 2, 3\}$ and $n = \{1, 2\}$, and draw a grid of those two sets. What does that tell you about multiplication? Do you see any similarities?

At the end of my counting, I have reached the number 6, so the product of 3 and 2 is 6.

This makes it clear that adding and multiplying can be done in such a way that they involving little more than counting. Hence, all of this can be done with the natural numbers (i.e., positive numbers), because those are the counting numbers.

30.2 Subtraction

Remark 30.3. The opposite of **addition** is **subtraction**. Addition is putting more onto a total tally, while subtraction is taking some away from a total tally.

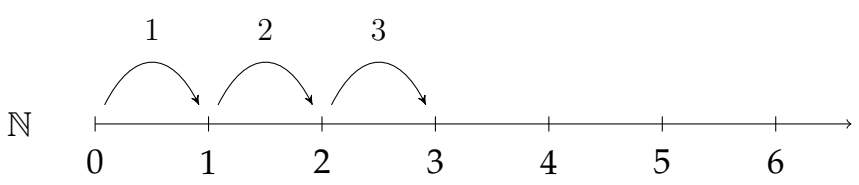
WHAT ABOUT THE OPPOSITE OF ADDITION? If addition is just adding more on, then the opposite is taking some away.

Why would we want to take things away? Well, to mention just one useful thing: it is easy to imagine that, at some point in human history, we realized we could handle bank accounts and credit better by giving people the ability to add and remove quantities from their accounts. So, I can **pay in** some amount (I can add to my balance), and I can **pay out** some amount (I can take away from my balance).

When we remove some, that's called **subtraction**. Consider subtracting n from m to get a difference p:

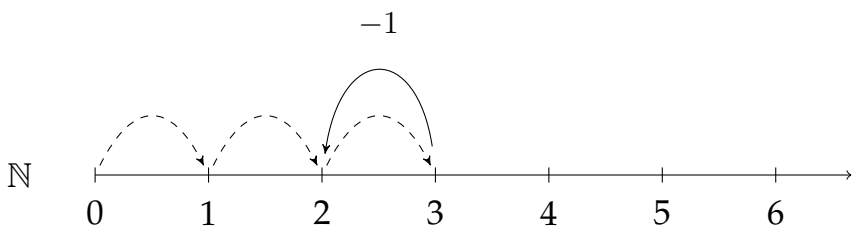
$$m - n = p$$

Intuitively, we can just think of this as taking away n items from a total tally of m. For instance, suppose I want to calculate $3 - 2$. First, I count up to 3:

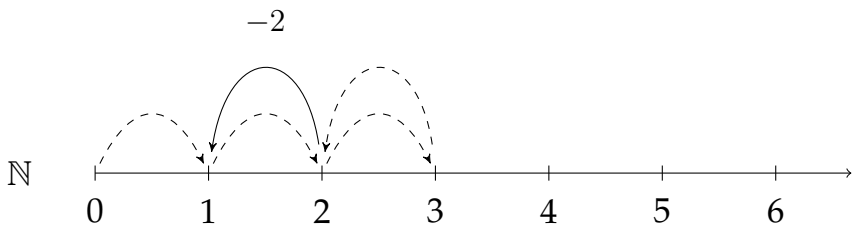


Terminology. To compute the answer to a problem with the form $m - n = p$, we take away n from m to get p. We call p the **difference** of m and n.

Next, I need to take away 2. So, I first take away one number (I substract 1):



Then I take away another number (now I am removing a total of 2):

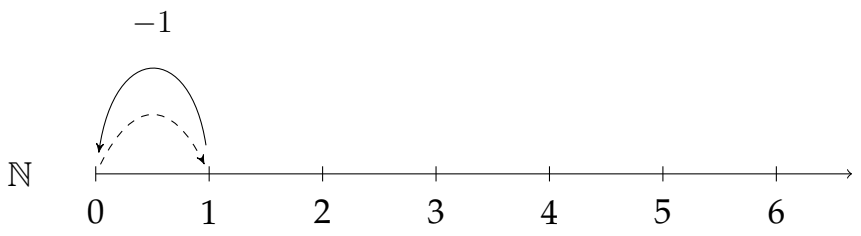


Remark 30.4. To signify that we are taking away 1, we can put a minus sign in front of the "1," like this: "-1." That symbolizes: "take away one." Likewise, "-2" symbolizes "take away two."

At the end of my counting, I have reached 1, so the difference between 3 and 2 is 1.

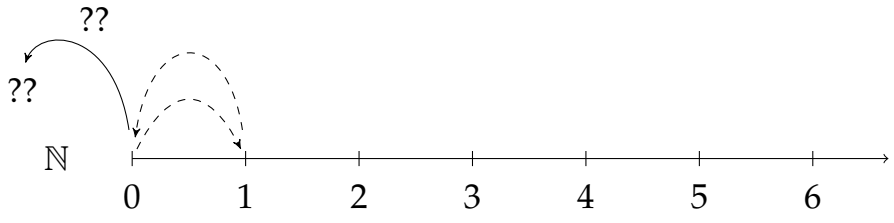
This is all well and good, but what happens if I want to take away more than I start with? For instance, how do I calculate $1 - 3$? To do that, I start by counting up to 1:

Then I need to take away 3. So I first take away 1 (i.e., I subtract 1):



Then, I need to take away two more. But where can I go from here? How do I take any more away?

Remark 30.5. We have gone back to the initial starting point (zero). How can we take away any more? There is nothing more to take away, because with the natural numbers, there are no **negative numbers**.



We have gone back to our starting point (zero), and so we can't take away any more. Hence, 1 - 3 actually can't be computed if we use the natural numbers!

If we really want to keep track of financial debts and other matters that involve subtraction, we need to include negative numbers, so that we can keep on going counting backwards when we need to go below zero.

30.3 The Integers

LET'S ADD THE NEGATIVE NUMBERS to our set of natural numbers. Here is the set of negative whole numbers:

$$\{\ldots, -3, -2, -1\}$$

Let's union that set with $\mathbb{N}$, so that we can combine those negative numbers and the natural numbers into one big set:

$$\{\ldots, -3, -2, -1\} \cup \mathbb{N}$$

Or, to write it out the members more explicitly:

$$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

Is this set identical to the natural numbers? No, it is not. Two sets are equal if they have the same elements, and this set has more elements than $\mathbb{N}$. So this is a *different* set. We call it the **integers**, and we denote it like this:

$$\mathbb{Z}$$

Read that aloud like so: "the set of integers" (or, if you like, you can just call it "Z").

So, the integers are defined as the set of natural numbers, plus all of the negative numbers too. Hence, we can write it out like this:

$$\mathbb{Z} = \{\ldots, -3, -2, -1\} \cup \mathbb{N}$$

Or, to write out the individual members explicitly:

$$\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$$

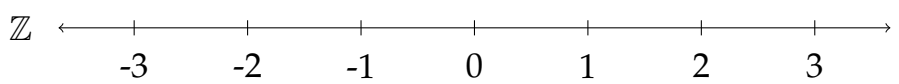
It is convenient to think of the integers as if they live evenly spaced on a number line that extends in both directions:

Terminology. We write each negative number by putting a minus sign in front of it. So "1" because "-1," "2" becomes "-2," and so on.

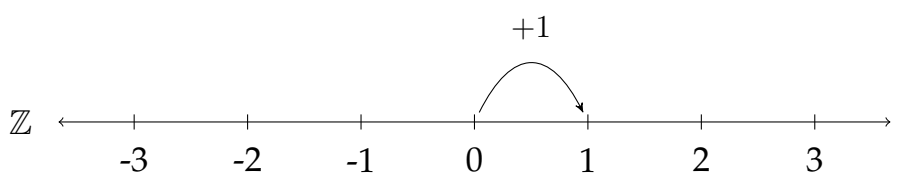
Remark 30.6. Recall from Chapter 9 that the **union** of a set A and a set B is the set we get when we combine all of the elements from both A and B.

Remark 30.7. Recall from Chapter 7 that two sets A and B are **equal** if every element of A is also in B and vice versa.

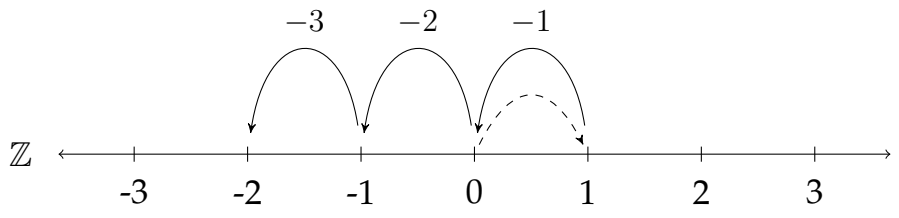
Remark 30.8. Recall from Chapter 7 that a set A is a **subset** of another set B if every element of A is also an element of B. Notice here that $\mathbb{N}$ is a subset of $\mathbb{Z}$, since every natural number $n \in \mathbb{N}$ is an element of $\mathbb{Z}$ too. Hence: $\mathbb{N} \subset \mathbb{Z}$.



Now it is possible to compute our earlier subtraction problem: $1 - 3$. I first count up to 1:



Then I remove 3 by counting backwards three tick marks:



At the end of my counting process, I have reached -2, so the difference between 1 and 3 is negative 2.

Notice an important fact about subtraction: it matters whether you are using the natural numbers, or the integers. For $\mathbb{Z}$, subtraction works as we all expect (we can go into the negative numbers). But for $\mathbb{N}$, subtraction does not always work.

Remark 30.9. When thinking about the integers on a number line, be careful: we must jump from each whole number to the next. We cannot move part way in between the tick marks, and get to (say) "two-and-a-half." The integers includes only whole numbers: two, three, negative three, and so on. There are no fractions or parts of numbers in between.

Remark 30.10. If there is a need to be clear, mathematicians often specify which set of numbers they are operating on. It is common to hear a mathematician say something like this, "think of addition on $\mathbb{N}$," or "let's take subtraction on $\mathbb{Z}$."

30.4 Summary

IN THIS CHAPTER, we learned about another set of numbers: the integers.

- The **integers**, which we denote as $\mathbb{Z}$, are defined as the union of the natural numbers and all the negative whole numbers. Hence: $\mathbb{Z} = \{\ldots, -2, -1\} \cup \mathbb{N}$.

- Hence, $\mathbb{Z}$ includes $\mathbb{N}$. That is, $\mathbb{N}$ is a subset of $\mathbb{Z}$: $\mathbb{N} \subset \mathbb{Z}$.
- Subtraction works as expected for $\mathbb{Z}$, but it cannot be done with $\mathbb{N}$ if the result goes below zero.

[31]

THE RATIONALS

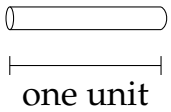
SUPPOSE WE ARE ANCIENT SURVEYORS. We are tasked with inventing a standardized way to measure distances (along straight lines). So, we need to be able to go to a line, and we need to be able to use our method to measure it, so that we can say it is such-and-such units long. How might we do this? How might we measure lines in a standard way?

Ponder. How might you measure distances, if you were an ancient surveyor? How would you establish a unit of measurement, and how would you use that to measure distances?

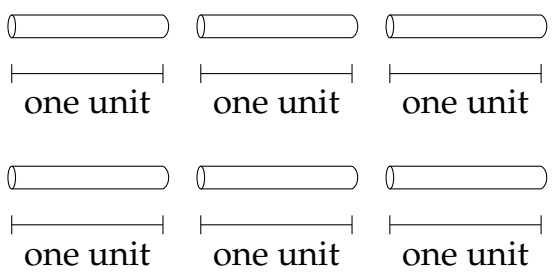
31.1 Measuring Distances

Here’s one way to measure distances. We find a straight stick, and say, “this stick here is our gold standard: the length of this stick will be what we mean by one unit of length.” Let’s call this our **unit stick** (because it has the length of one unit).

Remark 31.1. This is how the **yard** or **meter** stick came to be: someone picked a stick, they said, “this is how long a yard/meter will be,” and it was made official by the government. All other yard/meter sticks copy the length of that official original stick.

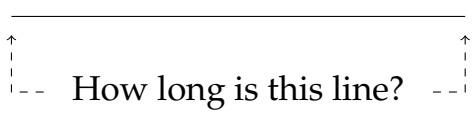


Next, we gather a bunch more sticks, and we cut them to precisely the same length as our unit stick. Now we have a bunch of unit sticks, all with the same length. Each stick has a length of one unit.



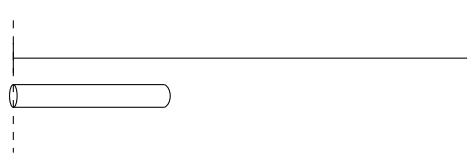
Remark 31.2. Once upon a time, a government might keep an official yard or meter stick around at the treasury. Other yard/meter sticks could always be checked for accuracy by comparing them to the official stick.

At this point, we can start measuring lines. We do that by lying down sticks, end to end, along the line. For example, suppose we want to measure this straight line:

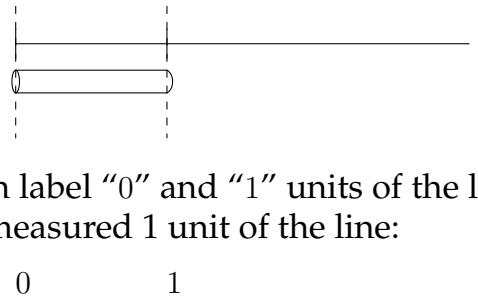


Remark 31.3. We can measure lines by using the measuring sticks we have built. The length of a "unit" doesn't really matter. What matters is that we all agree on how long one "unit" is, so that all units are the same length.

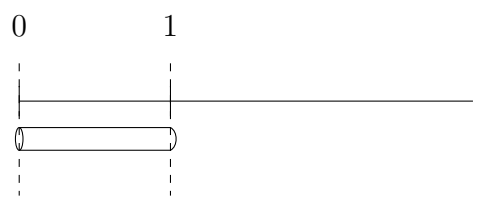
We can do it by lying down unit sticks, end to end. So first, we start at the left side of the line, and we lay down one unit stick, making sure it is lined up with the far left side of the line:



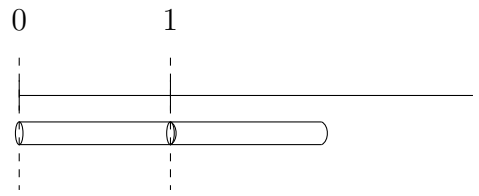
Next, we put a mark on the line at the point where our unit stick comes up to, to indicate that this mark is one unit:



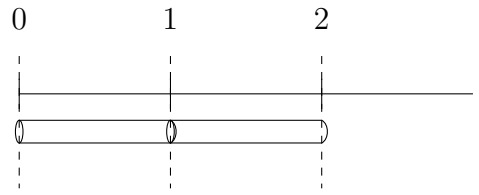
Now we can label "0" and "1" units of the line, to indicate that we have measured 1 unit of the line:



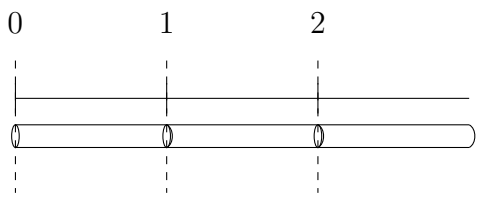
Next, we lay down a second unit stick, at the end of the first one:



Then, we draw a mark on the line again, which we can label as "2" units:



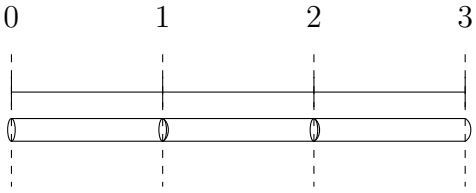
Then we can add a third stick:



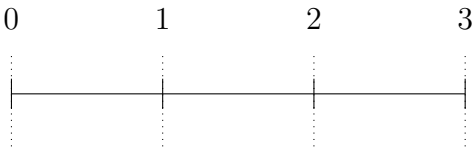
Remark 31.4. The "0" indicates where the measuring starts. It indicates that we have measured no amount of the line yet. The "1" indicates that we have measured one unit stick length of the line after "0." In other words, it is like a tally mark: i.e., one unit stick's length longer than **nought**.

Remark 31.5. Note that "2" here denotes that we have measured two unit stick lengths of the line. In other words, "2" is just a symbol for two tally marks: i.e., one unit stick's length longer than one unit stick's length longer than nought. This is just like counting with the **natural numbers**, except we are counting in stick lengths.

And of course, we can make a mark on our line again, to indicate that we have measured 3 units of the line:



With that, we have finished measuring the line. In essence, we have counted how many units there are in this line, by laying down our unit sticks end to end, and counting how many unit sticks we had to put down. And indeed, we can see that this line has 3 units in it:



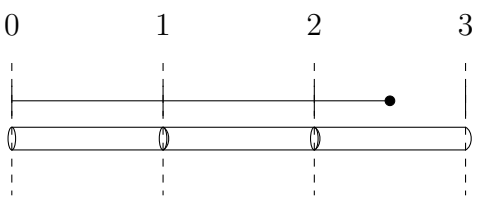
Remark 31.6. We can remove the sticks, and think about this picture more abstractly. To do that, we just think in terms of lines and tick marks. In our mind, we strip away the physicality of the sticks and the line, and just think about lengths appended to each other. This is another example of the process of **abstraction**: we strip away details that don't matter, in order to get to the essence of the matter.

31.2 Parts of Sticks

THIS WORKS GREAT if the lines we want to measure have ends that line up with an exact number of unit sticks. In our last example, the line measured out to 3 unit sticks, *exactly*.

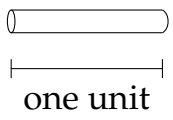
But what happens if the line doesn't end up so perfectly at the end of a unit stick? For example, consider this line:

Let's lay down our unit sticks, end to end, to see how this line measures up against our unit sticks:

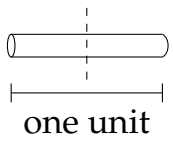


Uh oh. We put down three unit sticks, and the end of the line doesn't match up with the ends of any of our sticks. Rather, the line ends somewhere between the 2nd and 3rd unit stick. So, it's longer than 2 unit sticks, but it's shorter than 3 unit sticks.

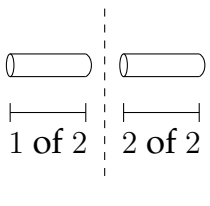
With our current system of using unit sticks to measure lines, we can't say precisely how long this line is. How might we remedy this problem? One option is that we can cut some of our unit sticks up into equal parts. For example, let's take a unit stick:



Then, let's cut it down the middle:



And that leaves us with two equal parts:

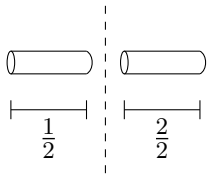


Notice that we labeled these as "1 of 2" and "2 of 2." That is because the first is one of the two parts, and the second the

Ponder. What do we do when the line doesn't correspond to an **exact** number of unit sticks? What do we do when the line goes only **part way** up a unit stick?

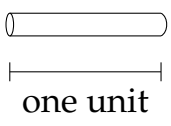
Remark 31.7. We can break up a unit stick into two equal parts. We call these **halves**. The first half-piece is the first half (or "the 1st out of 2" pieces), and the second half-piece is the second half (or "the 2nd out of 2" pieces). Do you recognize how "1 out of 2" corresponds to "$1/2$," or even "$\frac{1}{2}$"? What is the **ratio** of parts to whole here?

second of the two parts. For convenience, let's write "1 of 2" as "$1/2$," or even "$\frac{1}{2}$." Likewise, let's write "2 of 2" as "$2/2$," or even "$\frac{2}{2}$." Like this:



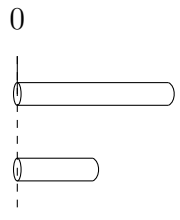
Let's call each of these pieces a **half-unit** stick, since we made them by cutting one unit stick into two halves.

We can measure the length of a **unit stick**, using **half-unit** sticks. For instance, we start with our unit stick:

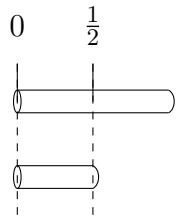


Ponder. Having smaller pieces is convenient, because we can use them to measure the length of a unit stick too. However, we made these smaller sticks by breaking up a unit stick into equal pieces. So, will we ever have a situation where these smaller pieces don't add back up to one unit?

Next, we put down one half-unit stick, and we line up the left sides of the two sticks. We mark "0" on the unit stack, to indicate where the measuring starts:

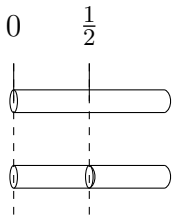


Then we mark "$\frac{1}{2}$" at the point on the unit stick where the half-unit stick comes up to:

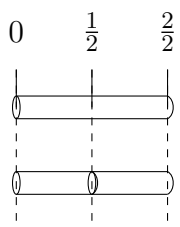


Remark 31.8. The "0" mark indicates that we have measured no units of length yet, and the "$\frac{1}{2}$" mark indicates that we have measured one half of a unit stick's length past nought. So, it's like we're counting with natural numbers again, except we're counting with halves.

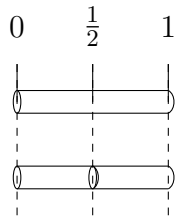
Then, we add another half-unit stick at the end of the first one:



And we mark the unit stick with "$\frac{2}{2}$" at the point where the second half-unit stick ends:



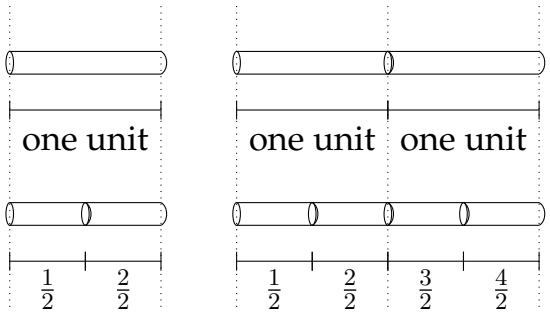
So, we have used our half-unit sticks to measure the unit stick, and we can see that the length of one unit stick is two half-unit sticks. Since "$\frac{2}{2}$" (i.e., 2 out of 2 half-units) comes up to one full unit, so we can erase the "$\frac{2}{2}$" label, and just write "1," like this:



Hence, these two half-unit sticks, when combined, measure up to one unit. Likewise, we could go on and add more half-unit sticks, to count longer portions. For instance, we can count 4 half-unit pieces — i.e., $\frac{1}{2}$ (one half-unit stick), $\frac{2}{2}$

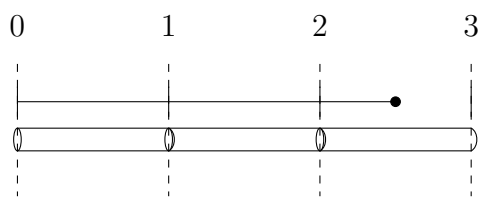
Notation 31.1. Notice the role of the two numbers in our notation "$\frac{n}{m}$." The bottom number m indicates **how many equal pieces** we have broken a unit stick into, and the top number n is just a **natural number**, i.e., it is just a counter indicating how many of the $\frac{1}{n}$ sized pieces we have counted. Here we can see this in action. The bottom number in these ratios is 2, which means we have broken up the unit stick into 2 equal pieces. The top number in these ratios indicates how many of these pieces we have counted up to. We start with "0" to indicate nought (we could also write "$\frac{0}{2}$" to indicate that we have counted no half-units yet). Then we count to "$\frac{1}{2}$," to indicate that we have counted 1 half-unit. Finally, we reach "$\frac{2}{2}$," to indicate that we have counted 2 half-units

(two half-unit sticks), $\frac{3}{2}$ (three half-unit sticks), and $\frac{4}{2}$ (four half-unit sticks) — which adds up to 2 unit sticks:

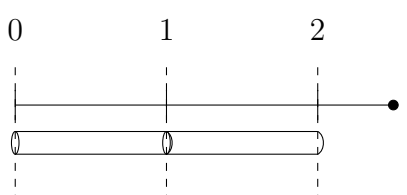


Remark 31.9. It is easy to see from these pictures that we can keep adding half-unit sticks on, one at a time, to measure distances just as we do with unit sticks. We can represent this in our ratio notation too. For each tick mark, we write a "2" on the bottom to indicate that we are counting in half-unit sticks, and then we just increment the number on the top by one each time: $\frac{0}{2}$, $\frac{1}{2}$, $\frac{2}{2}$, $\frac{3}{2}$, ..., $\frac{1334}{2}$, ... We can go on forever like this. Of course, we can also see that $\frac{0}{2}$ measures the same distance as 0 unit sticks, $\frac{2}{2}$ the same as 1 unit stick, $\frac{4}{2}$ the same as 2 unit sticks, $\frac{110}{2}$ the same as 55 unit sticks, and so on.

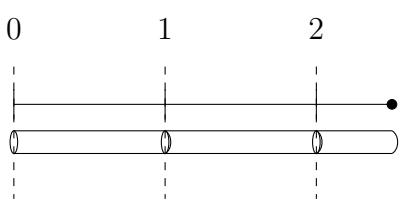
Now we can return to the line we were trying to measure before, which fell somewhere between 2 units and 3 units long. Here is the picture again:



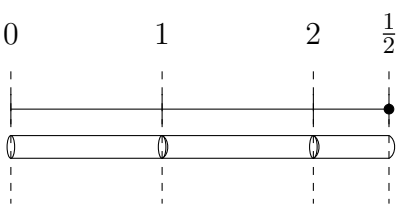
We can measure this line by taking away the third unit stick and keeping just two unit sticks:



And then we can add a half-unit stick to the end:

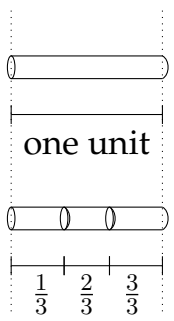


Now we can mark that we have 2 full units, and one half-unit:



31.3 More Parts

WE CAN DIVIDE UNIT STICKS into more and more equal parts. Above, we divided our unit stick up into **two** parts (halves). But we can also divide a unit stick up into **three** parts (thirds):



We could divide it up into fourths, fifths, sixths, and so on. In principle, there is no end to how many pieces we can divide a unit stick into. Of course, we are limited by how small our tools will allow us to go, but that is just a physical limitation.

There is no reason why we can't imagine dividing up unit sticks into smaller and smaller pieces, forever. We can imagine breaking up a unit stick into a hundred pieces:

Remark 31.10. We can divide a unit stick up into any number of equal parts (assuming our tools are precise enough to handle really small pieces). So, we can divide one unit up into 2 equal pieces, or 3 equal pieces, or 4 equal pieces, and so on forever. It may be hard to imagine, but we could even divide it up into a million equal pieces, a million and one equal pieces, and so on, forever.

Notation 31.2. When we divide a unit stick up into 2 pieces, we put 2 on the bottom of the ratio $\frac{n}{m}$. Hence, we have $\frac{1}{2}$, $\frac{2}{2}$, and so on. When we divide a unit stick up into 3 pieces, then we put 3 on the bottom of the ratio. Hence, we have $\frac{1}{3}$, $\frac{2}{3}$, and so on. Likewise, if we divide the unit stick up into 7 pieces, we put 7 on the bottom of the ratio (which gives us $\frac{1}{7}$, $\frac{2}{7}$, etc).

$$\frac{1}{100}, \frac{2}{100}, \frac{3}{100}, \dots$$

We can imagine breaking it up into a million pieces:

$$\frac{1}{1,000,000}, \frac{2}{1,000,000}, \frac{3}{1,000,000}, \dots$$

And so on. We have hit on a special idea here: we can break up a unit into equally sized smaller pieces, and we can use those to measure smaller portions of a unit length.

31.4 The Rational Numbers

IN DOING ALL THIS, we have encountered a new kind of number: we encountered **parts** or **portions** of numbers. Why is this "new"?

Well, with $\mathbb{N}$ and $\mathbb{Z}$, we only have **whole** numbers. If we look in $\mathbb{N}$ and $\mathbb{Z}$, and look around, we will see numbers like 1, 2, -7, 253, and so on. But we will never encounter half of a number, or a third of a number. That is why we say $\mathbb{N}$ and $\mathbb{Z}$ contain **whole numbers**. Those numbers are whole units, which are not broken up into smaller parts.

Terminology. The natural numbers $\mathbb{N}$ and the integers $\mathbb{Z}$ are **whole numbers**, because each one is a whole unit that is not broken up into smaller pieces.

But now we have seen *parts* of units, i.e., halves, thirds, hundreths, and so on. Let us call each such part a **ratio** (because $\frac{1}{100}$ is a ratio of 1 out of a 100 parts). Another name we will use is a **fraction** (because $\frac{1}{100}$ is a fraction of a whole).

Terminology. A **ratio** (or synonymously: a **fraction**) indicates some amount of equally sized *parts* or *portions* of a whole unit. For instance, $\frac{2}{3}$ indicates that we have 2 of 3 equally sized pieces of a unit, and $\frac{5}{3}$ indicates that we have 5 such third-sized pieces.

Let us put all of these ratios (synonymously: fractions) into a set. What exactly goes into this set? Let's build up this set slowly, so we can see all the fractions that go into it. Let's start with 0 in our set:

$$\{0\}$$

Next, let's add the halves:

$$\{0, \frac{1}{2}, \frac{2}{2}\}$$

However, there are many more halves beyond this. There is $\frac{3}{2}$ (i.e., 3 half-unit sticks), and $\frac{4}{2}$ (i.e., 4 half-unit sticks), and so on forever:

$$\{0, \frac{1}{2}, \frac{2}{2}, \frac{3}{2}, \frac{4}{2}, \frac{5}{2}, \dots\}$$

Note that we already have an infinitely large set of fractions here, just with the halves. But we're not done. Next, we need to add the thirds. And in the same way, there is an infinite number of thirds:

$$\{0, \frac{1}{2}, \frac{2}{2}, \frac{3}{2}, \frac{4}{2}, \frac{5}{2}, \dots, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}, \frac{4}{3}, \frac{5}{3}, \dots\}$$

We should next add the fourths:

$$\{0, \frac{1}{2}, \frac{2}{2}, \frac{3}{2}, \frac{4}{2}, \frac{5}{2}, \dots, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}, \frac{4}{3}, \frac{5}{3}, \dots, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, \frac{4}{4}, \frac{5}{4}, \dots\}$$

You can see how we can keep going, to build up this set. This set keeps going, in the positive direction.

We also need to go in the **negative** direction too. Hence, we need to add the negative halves, i.e., $-\frac{1}{2}$, $-\frac{2}{2}$, $-\frac{3}{2}$, and so on:

$$\{\dots, -\frac{3}{2}, -\frac{2}{2}, -\frac{1}{2}, 0, \frac{1}{2}, \frac{2}{2}, \frac{3}{2}, \dots, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}, \frac{4}{3}, \dots\}$$

And we need to add all of the negative thirds:

Remark 31.11. It would be impossible for any human to list out all the halves, because it just keeps going. For any $\frac{n}{2}$, we can always add another: $\frac{n+1}{2}$! Just increment the top number by one! So, we have to imagine that this list of halves has already been built (by God, perhaps).

Remark 31.12. It might be clearer to list this out in rows and columns. Something like this:

$$\begin{array}{cccccc}
\frac{0}{2}, & \frac{1}{2}, & \frac{2}{2}, & \frac{3}{2}, & \frac{4}{2}, & \frac{5}{2}, \cdots \\
\frac{0}{3}, & \frac{1}{3}, & \frac{2}{3}, & \frac{3}{3}, & \frac{4}{3}, & \frac{5}{3}, \cdots \\
\frac{0}{4}, & \frac{1}{4}, & \frac{2}{4}, & \frac{3}{4}, & \frac{4}{4}, & \frac{5}{4}, \cdots \\
\frac{0}{5}, & \frac{1}{5}, & \frac{2}{5}, & \frac{3}{5}, & \frac{4}{5}, & \frac{5}{5}, \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$

We can see how each row keeps expanding forever to the right, and each column keeps expanding forever downwards.

Remark 31.13. With negative fractions, the rows expand forever both left and right:

$$\begin{array}{ccccccccc}
 & & -\frac{2}{2}, & -\frac{1}{2}, & \frac{0}{2}, & \frac{1}{2}, & \frac{2}{2}, & \frac{3}{2}, & \frac{4}{2}, \\
\cdots, & & -\frac{2}{3}, & -\frac{1}{3}, & \frac{0}{3}, & \frac{1}{3}, & \frac{2}{3}, & \frac{3}{3}, & \frac{4}{3}, & \cdots \\
 & & -\frac{2}{4}, & -\frac{1}{4}, & \frac{0}{4}, & \frac{1}{4}, & \frac{2}{4}, & \frac{3}{4}, & \frac{4}{4}, \\
\cdots, & & -\frac{2}{5}, & -\frac{1}{5}, & \frac{0}{5}, & \frac{1}{5}, & \frac{2}{5}, & \frac{3}{5}, & \frac{4}{5}, & \cdots \\
 & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$

$$\{\ldots, -\frac{4}{3}, -\frac{3}{3}, -\frac{2}{3}, -\frac{1}{3}, \ldots, -\frac{3}{2}, -\frac{2}{2}, -\frac{1}{2}, 0, \ldots\}$$

You can see that this is a very large set of fractions. In both the positive and negative directions, we have an infinite amount of halves, and an infinite amount of thirds, and an infinite amount of fourths, and so on, forever!

We call this set the **rational numbers**. They are not called "rational" because they are smart or anything like that. The name "rational" is used because it is the adjective form of **ratio**. Hence these numbers are just *ratios*, i.e., *fractions*. We use the letter "$\mathbb{Q}$" to denote them. Hence, we can write our set out like this:

$$\mathbb{Q} = \{\ldots, -\frac{2}{3}, -\frac{1}{3}, \ldots, -\frac{2}{2}, -\frac{1}{2}, 0, \frac{1}{2}, \frac{2}{2}, \ldots, \frac{1}{3}, \frac{2}{3}, \ldots\}$$

31.5 Summary

IN THIS CHAPTER, we learned about the rational numbers, i.e., ratios or fractions.

- $\mathbb{N}$ and $\mathbb{Z}$ contain only **whole numbers**.
- If we break whole numbers up into parts (i.e., halves, thirds, fourths, fifths, and so on), then we get **fractions**, or synonymously **ratios**.
- The **rational numbers**, which we denote as "$\mathbb{Q}$," is the set of all positive and negative fractions.
- Like the natural numbers and the integers, $\mathbb{Q}$ is an **infinite** set. There is no end to the numbers it contains.

[32]

THE REAL NUMBERS

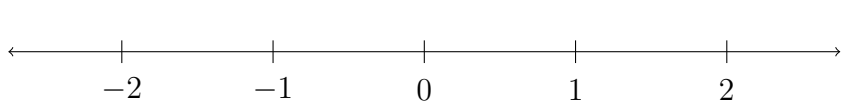
THERE AREN'T ENOUGH RATIONAL NUMBERS to cover all the points on a number line. So, if we want to calculate with precision, we need a better set of numbers than the rationals. In this chapter, we will look at the set of **real numbers**, which includes every possible point on the number line.

Ponder. Can you think of any circumstances in which the rational numbers wouldn't be good enough to use for calculations? When are fractions not good enough?

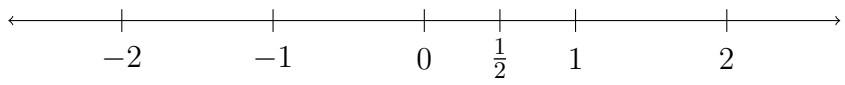
32.1 Gaps between Fractions

AS WE SAW IN CHAPTER 31, the rational numbers is the set of all fractions. We can construct fractions by imagining that we divide a unit stick up into smaller, equally-sized pieces. In theory, there is no end to how small we can make the pieces. We can keep dividing each unit infinitely.

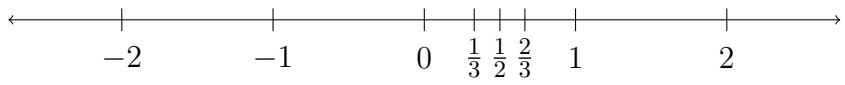
However, no matter how tiny we make our pieces, there will *always* be **gaps** between them. Imagine a number line with the integers on it, e.g.:



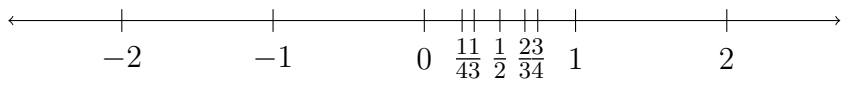
Now imagine adding tick marks for fractions. For instance, let's add tick marks for the halves that occur between 0 and 1:



Notice that there are gaps between the tick marks. Let's now add tick marks for the thirds that occur between 0 and 1:



Notice that there are still gaps between these tick marks. Let's add tick marks for the fourths that occur between 0 and 1:



Again, we still have gaps between our tick marks. No matter how many fractions we add between 0 and 1, there will always be gaps between them. Of course, we might have to zoom in to see the gaps once we get really, really small, but the idea is clear enough: there will always be gaps between the tick marks.

Ponder. We can always divide a unit into more fractions. If we've divided it into fourths, we can then divide it into fifths. If we've divided it into $n$ths, we can divide it into $(n+1)$ths! With that in mind, do you think this means that there *must* be a gap between every nth? Otherwise, how could we divide it further?

32.2 The Real Number Line

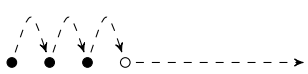
IF WE WANT MORE PRECISE MEASUREMENTS for things like lines, or even particles moving about in space, we can't have gaps between our numbers. We need a set of numbers that runs *continuously* through the gaps.

How do we model this? Let's imagine a particle flying through space in a straight line. Let's draw it as a dot, with a trajectory, something like this:

Now, let's suppose that the particle starts to move forward along its trajectory. As it moves, it passes through points on its trajectory. Let's draw the points with solid black dots. So, for instance, it moves forward a tiny bit, leaving behind a dot where it's been:

Then it moves some more, leaving behind more black dots:

However, this drawing isn't good enough, because we can see tiny spaces between the dots. This isn't right, because it's not like the particle "jumps" from one point to the next. If we zoom in real close, we won't see this:



No, if we zoom in real close, we'll see more dots that the particle has passed through:

Terminology. In this context, to go **continuously** through the points means to pass through the points without any gaps. You might imagine putting your pencil on the paper, and drawing a line without lifting your pencil. If you lifted your pencil and put it down again, that would leave a "gap" in the line, and so that line would not be continuous.

And if we zoom in even more, we'll see more dots:

And even this drawing isn't good enough, because there are still gaps. It really needs to be a solid black line, packed infinitely densely with points:

Ponder. Try to imagine all the points that the particle travels through as it moves in a straight line through space. Remember that the particle never "jumps" or "warps" from one spot in space to another, later spot. What do you see in your mind when you try to imagine all these points?

This is because the particle travels continuously along its trajectory, and there is *never* a gap in its movement. It never skips or "jumps" from one point in space to another point in space farther along. Between any two points x and y on the trajectory, no matter how infinitesimally close they may be, there are still infinitely many more points between which the particle passes through as it travels from x to y.

If we want to really capture all of the points on a line, we need to imagine something like the sequence of points that the particle travels through. We need to say that we have infinitely many points, packed in so densely that there are literally no gaps anywhere. Between any two points x and y, there are infinitely many more points.

Terminology. The **real number line** is an imaginary model of a real line that extends in space. A real line has no gaps, so in order to imagine this as a set of points, we need to imagine infinitely many points packed infinitely densely into a straight line.

So, let's imagine this infinitely dense set of points, all arranged in a straight line. This is how we can model a *real* line in our minds. This is how we can capture the idea that a solid line extended in space has no gaps (for if it had gaps, it wouldn't be a solid line). Let's call this the **real number line**.

32.3 The Real Numbers

LET'S TAKE ALL THE POINTS that we just imagined, arranged in a line, and let's put them into a set. We call this set the **real numbers**, and we denote it like this:

$\mathbb{R}$

The real numbers are a truly spooky collection of objects. The sheer size and density of them is mind boggling.

It's nice to have so many points, but we need names for them all. So how do we get names for all of them?

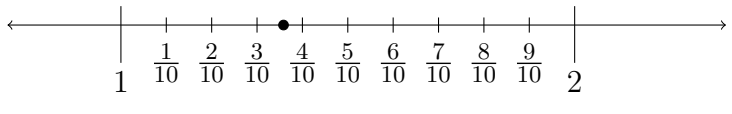
The established tradition is to use decimal points. Let's say we have a point on the real number line that lies somewhere between 1 and 2:

What is the name of this object, using a decimal point? We know it is bigger than 1, so it will be one-point-something:

$$1.??????$$

The question marks are empty slots that we need to fill in. When we fill in these slots with numbers, we say that this is a **decimal expansion**. Think of a decimal expansion as the process of adding in numbers after the decimal point.

How do we calculate the numbers that go into our decimal expansion? First, we divide the portion of the line between "1" and "2" into 10ths, like this:



We can see that our dot falls after $\frac{3}{10}$, so we write 3 in the first empty slot in our decimal expansion:

$$1.3?????$$

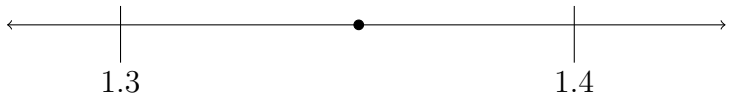
Next, we zoom in on the area marked $\frac{3}{10}$ and $\frac{4}{10}$, which are 1.3 and 1.4 on the line:

Terminology. The **real numbers** are all the points that fall on the real number line. We denote the set like this: $\mathbb{R}$. It is an infinitely dense set of objects.

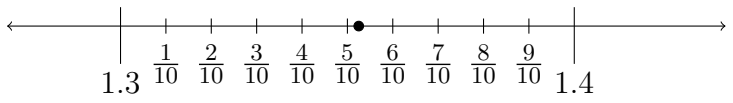
Ponder. Suppose you had to make an educated guess at where this point lies. Is it 1.5? 1.25? 1.33? Somewhere in between? How might you guess the first couple of decimal places?

Terminology. A **decimal expansion** has the form $n.d_1d_2d_3\ldots$, where n is a positive or negative integer, and d_1, d_2, d_3, and so on are replaced by digits.

Remark 32.1. We could write these fractions as decimal numbers if we were so inclined. For instance, we could write 1 and $\frac{1}{10}$ as 1.1, we could write 1 and $\frac{2}{10}$ as 1.2, and so on.



Next, we again divide this portion of the line up into 10ths, like this:

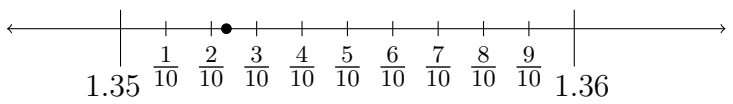


We can see that our dot is after $\frac{5}{10}$, so we write 5 in the next empty slot in our decimal expansion:

$$1.35?????$$

We can repeat this process again. We zoom in on the area marked $\frac{5}{10}$ and $\frac{6}{10}$, which are 1.35 and 1.36 on the line:

And we again divide this portion of the line into 10ths, like this:



We can see that our dot falls on the line after the $\frac{2}{10}$ mark, so we write 2 in the next empty slot in our decimal expansion:

$$1.352????$$

We can keep going like this forever, always zooming in, finding the next closest decimal digit in our decimal expansion.

Of course, the more times we do this, the more digits we add to our decimal expansion. And the more digits we add,

Remark 32.2. We could write these fractions as decimal numbers too, if we were so inclined. For instance, we could write 1.3 and $\frac{1}{10}$ as 1.31, we could write 1.3 and $\frac{2}{10}$ as 1.32, and so on.

Remark 32.3. We can write these fractions as decimal numbers too, just as before. For instance, we can write 1.35 and $\frac{1}{10}$ as 1.351, we can write 1.35 and $\frac{2}{10}$ as 1.352, etc.

the more accurate our decimal expansion becomes at representing the exact point on the line where our dot falls.

In truth, a real number has an **infinite number of digits** in its expansion. It's not just 1.352. It's $1.352\ldots$, with digit after digit appearing at the end forever. In practice, we obviously can't keep expanding (we'd run out of time). In practice, we usually just stop after we have reached a level of precision that is good enough for our purposes.

For instance, with money, we don't need to expand beyond two decimal places, but with a sensitive physics experiment, maybe we would want to expand more digits, so that we can be more accurate.

Remark 32.4. Every **real number** has an **infinite number of digits** in its decimal expansion. Hence, any decimal number with a finite number of digits is just an **approximation** of the real number. The more digits we add to the expansion, the more **accurate** the approximation.

32.4 How Other Numbers Fit

FOR SIMPLICITY, we can think of the real numbers as just all numbers with decimal points. For instance, these would all be real numbers:

$$1.0 \qquad 2.457 \qquad 3.14159\ldots \qquad 32,754.9879879999\ldots$$

We sometimes write only one or two digits in these numbers' expansions, even though in truth, every real number actually has an infinite expansion.

Every **natural number** can be written as a real number, by adding infinitely many zeros afterwards:

$$\begin{aligned} 0 &\mapsto 0.00000000\ldots \\ 1 &\mapsto 1.00000000\ldots \\ 2 &\mapsto 2.00000000\ldots \\ \vdots &\quad \vdots \end{aligned}$$

The **integers** can be written similarly:

$$\begin{aligned}
-2 &\mapsto -2.00000000\ldots \\
-1 &\mapsto -1.00000000\ldots \\
0 &\mapsto 0.00000000\ldots \\
1 &\mapsto 1.00000000\ldots \\
2 &\mapsto 2.00000000\ldots
\end{aligned}$$

Remark 32.5. In practice, we usually don't write all the zeros. We don't write $2.0000000\ldots$ We just write 2.0 or even 2. And we don't write $0.75000000\ldots$ We just write 0.75.

And fractions (the **rational numbers**) can of course be written as decimals too, with infinitely many zeros in the expansion. For instance, here are some fourths:

$$\begin{aligned}
{}^{0}\!/_{4} &\mapsto 0.00000000\ldots \\
{}^{1}\!/_{4} &\mapsto 0.25000000\ldots \\
{}^{2}\!/_{4} &\mapsto 0.50000000\ldots \\
{}^{3}\!/_{4} &\mapsto 0.75000000\ldots \\
{}^{4}\!/_{4} &\mapsto 1.00000000\ldots \\
{}^{5}\!/_{4} &\mapsto 1.25000000\ldots
\end{aligned}$$

This makes it clear that the real numbers include all of the natural numbers, all of the integers, and all of the rational numbers. They are all subsets of the real numbers. To put it in symbols:

$$\begin{aligned}
\mathbb{N} &\subseteq \mathbb{R} \\
\mathbb{Z} &\subseteq \mathbb{R} \\
\mathbb{Q} &\subseteq \mathbb{R}
\end{aligned}$$

Remark 32.6. Recall that we denote the natural numbers as $\mathbb{N}$, we denote the integers as $\mathbb{Z}$, and we denote the rationals as $\mathbb{Q}$.

32.5 Summary

IN THIS CHAPTER, we learned about the real numbers, i.e., the set that includes every single point on the real number line.

- The **real number line** models a real line extending in space. A real line has no gaps. To capture this, we say that the real number line is a collection of points arranged in a straight line, and between any two points x and y, there are infinitely many more points.
- If we take all these points and put them into a set, we call that set the **real numbers**, and we denote it like this: $\mathbb{R}$.
- To generate a name for any of the object in $\mathbb{R}$, we can create a **decimal expansion**. To do that, we add a decimal point, and then we add digit after digit. With each digit, we get a tenth of the way closer to where the point really lies on the real number line.
- However, each real number has an **infinite number of digits** in its decimal expansion (even if those digits are all zeros). Hence, in practice we stop at whatever number of digits **we need** to suit our purposes.

[33]

FURTHER READING

To pursue numbers further, the following list may offer some helpful starting points.

- Cummings (2018, ch. 1 and appendix A) provide an introduction to the real numbers and what they can be used for, as well as a discussion of how to construct the natural numbers, integers, rational numbers, and real numbers.

- Stewart (1995, chs. 6, 16) offers some beginner-level discussion of what the real numbers are used for.

- Stewart and Tall (2015, chs. 2, 9, 10) offer a fairly thorough introduction to many of the issues surrounding the real numbers.

- Warner (2019, ch. 5) provides an excellent introduction to the natural numbers, integers, rational numbers, and real numbers. Warner constructs each set of numbers fully, and then defines addition, multiplication, and ordering on them. In doing so, Warner shows how all of these sets are built from the ground up.
- Madden and Aubrey (2017, chs. 2, 3, 4, 6, and 14, 15) offer thorough coverage on the natural numbers, integers, rationals, and real numbers.

Part [7]

INFINITIES

How big is infinity, exactly? And what exactly do we mean by "infinity"? Are some infinite collections bigger than other infinite collections? Can we even compare the size of different infinite collections?

In Part 7, we discuss the idea of infinity as the size of an infinitely large collection. We define a way to **compare** two infinitely large sets without counting them, so that we can check if they are the same size or not. We define what it means for an infinite set to be **listable** vs. **unlistable**, and we look at some examples.

[34]

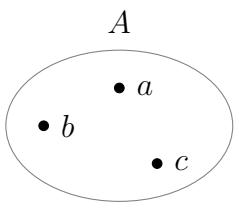
MEASURING SIZE

Ponder. How do you measure the size of a collection? Do you count it? Or do you compare it to another collection whose size you do know? Can you think of a way to figure out if infinitely large collections are the same size?

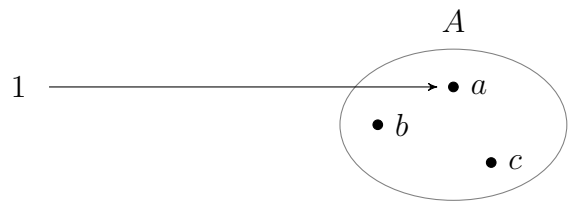
HOW BIG IS INFINITY? Is there even a way to make the idea of infinity and "how big" it is precise? Are some infinitely large collections bigger than others? In this chapter, we look at how we can **compare** the size of infinitely large collections.

34.1 Size by Counting

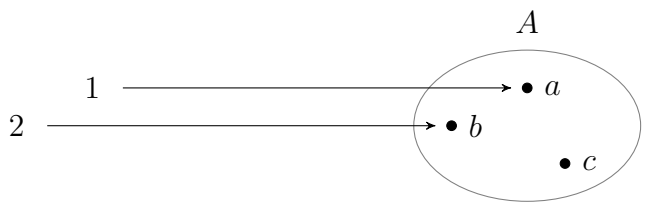
HOW DO WE FIGURE OUT THE SIZE of a set? One way to do it would be to just sit down and count how many objects it has. For instance, suppose I have a set A that looks like this:



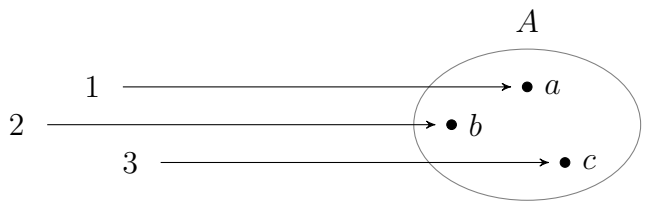
To count this, I would point to one element, and say "there's one:"



Then I would point to another element and say, "there's two":



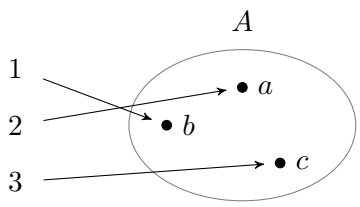
Then I would point to the last element and say, "there's three":



With that, I have finished counting, and so I can conclude that the **size** of A is 3. To use the technical terminology, its **cardinality** is 3:

$$|A| = 3$$

Remark 34.1. Notice that I could count these objects in a different order, and I'd get the same answer. For instance:



Remark 34.2. Recall from Chapter 8 that the size of a set is how many objects it contains. We call the size of a set its **cardinality**. To denote the size of a set B, we write "$|B|$." To say that the size of B is 10 (i.e., to say that B contains 10 objects), we write "$|B| = 10$."

34.2 What about infinity?

IF WE HAVE TO COUNT the elements in a set to figure out its cardinality, what happens when we try to count the elements in an **infinite** set?

As an example, let's take the positive integers. Let's call this set $\mathbb{Z}^+$, and say that it includes just the positive integers. Hence it is this:

$$\mathbb{Z}^+ = \{1, 2, 3, \dots\}$$

Suppose we want to determine the cardinality (size) of this set. Can we do it by counting it? Well, if we tried that, we would start at 1, and we would count that. Then we would go to 2, and we would count that. Then we would go to 3, and we would count that. We would keep going. And going. And going.

We would never finish our task of counting, since $\mathbb{Z}^+$ keeps going forever. And if we never finish counting, we will never arrive at any particular **size** for the set we're trying to count.

The moral of the story here is that if we want to calculate the **cardinality** of a set, then counting works fine for **finite sets**. However, it does not work for **infinite sets**. So how do we handle the size of infinite sets?

34.3 Infinity Is Not a Number

WE NEED TO BE CAREFUL HERE. We started by asking about the size of sets, and we looked at an example of a small set, containing only three elements. The size of this set was a particular number. In fact, it was 3.

Now, what number is this exactly? Since we counted up to 3, it must be a counting number (i.e., a natural number $\mathbb{N}$). To be more precise, it is the third counting number that comes after nought. In other words, it is $s(s(s(\oslash)))$, where "3" is just the convenient name we use for it. So:

$$s(s(s(\oslash))) \in \mathbb{N} \qquad \text{i.e.} \qquad 3 \in \mathbb{N}$$

Remark 34.3. Recall from Chapter 29 that "$s(s(s(\oslash)))$" is "the successor of the successor of the successor of nought," but since that's such a mouthful, we have a more convenient name for it: 3.

If you think about it, if we count how many items are in any **finite** set, we will always end up with a natural number. E.g., try to count a set with 5 items in it, or 13 items in it, or 345,234,023 items in it, and so on. In each case, we count up to a particular natural number.

So, we can say that the **size** of any finite set will be a *particular* natural number. It will always be a particular element from inside the set of natural numbers.

Let us set down a definition here, so we can be clear about what we mean. Let us agree to the following convention: when we say "a particular number," we mean an individual **element** from the set of the natural numbers.

Terminology. When we speak of a **particular number**, we mean some element from the set of natural numbers, i.e., some $n \in \mathbb{N}$.

Definition 34.1 (Particular Numbers.)**.** When we speak of **a particular number**, we mean an element n from the set of natural numbers, i.e., $n \in \mathbb{N}$.

With that in mind, let's turn to infinite sets. What about the size of infinite sets? What is the size of, say, $\mathbb{Z}^+$?

As we saw, we cannot ever *count* the size of $\mathbb{Z}^+$, because there will never be an end to the counting. No matter how big a number we count up to, there will always be one more after that that we could count up to, and then one more after that to count up to, and then . . . and so on, forever.

So, the size of $\mathbb{Z}^+$ cannot ever be a **particular number**. And remember, by "a particular number," we mean a member of $\mathbb{N}$. So, the size of $\mathbb{Z}^+$ is *not* a natural number.

Still, we want to be able to talk about the *size* of $\mathbb{Z}^+$, even though we can't use a natural number to do it. So, let's just invent a special name for this size. Let's call it **infinity**.

The word "infinity" is a **size**, not a **particular number**. As a tagline, let's just say this:

Infinity is not a number

This tells us that we need to think about infinite sets purely in terms of *size*, without bringing any counting into it.

Terminology. **Infinity** is a **size**, not a **particular number**. Specifically, it refers to the size of a never ending sequence (like $\mathbb{Z}^+$ or $\mathbb{N}$).

34.4 Comparing for Size

THERE IS A WAY to think about the sizes of sets, which doesn't involve doing any counting. We can simply **compare** two sets, and see if one is bigger than the other.

When we do this, we may not be able to say "how many" items there are in each set (because we aren't counting them), but we will be able to say things like "this set has *more* items than that one," or "these two sets have the *same* number of items."

The way we do it is this. We pair up the elements from the two sets, one by one, and we see if we can pair them all up. We take one from the first set and one from the second set, and pair those two off. Then we take another from the first set and another from the second set, and pair those two off. We keep going until we've paired off as many as we can.

When we get to the end of this, if there are any leftovers from one of the sets, then we know that set has *more* items

Remark 34.4. If we merely **compare** two sets against each other to see if one is bigger, we do not need to count them. We simply pair them up, side by side, and see if there are any extras.

- When a teacher looks out at the classroom, they scan the desks in the room, to see which desks are empty. If all desks are full, then there's the same number of students and desks.
- When you set the cutlery and plates out at the dinner table, you scan very quickly to see which plates are missing cutlery. If every plate has an accompanying cutlery, then you have an equal number of plates and cutlery sets.

than the other one. If we get to the end and there are no leftovers, then we know the two sets are exactly the *same* size.

34.5 Size by Isomorphism

THINK ABOUT THE MECHANISM OF PAIRING that we just described. When we try to match up the items in two sets in an exact, one-to-one, reversible way, what do we call that? It's a **bijective function** (or **isomorphism**).

Bijective functions can only be constructed between sets that are equal in size. So, if we can construct a bijective function between two sets, then we can conclude that they must be the same size.

Let's put this down as a definition. Let's say that two sets are the **same size** (or synonymously, **equinumerous**) if we can construct a **bijective function** between them.

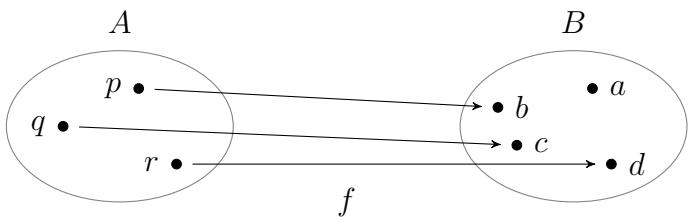
Remark 34.5. Recall from Chapter 16 and Chapter 17 that a **bijective function** is an injective and surjective function, and it can only be constructed between two sets that are exactly the same size. If we can build a bijective function between two sets, then that is an **isomorphism**. It tells us that the two sets are exactly the same "under the hood," so to speak. They differ only in the surface names of their elements.

Terminology. Two sets A and B are **equinumerous** if they have the same **cardinality** (size). We denote this by writing "$|A| = |B|$."

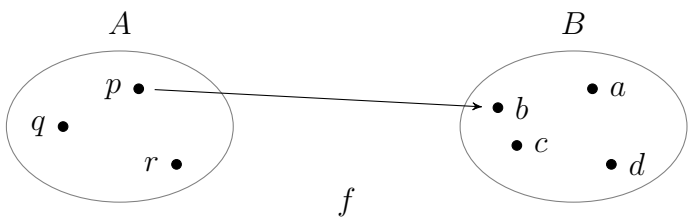
Definition 34.2 (Equinumerous sets)**.** For any two sets A and B, we will say that A and B are **equinumerous** if we can construct a bijective function $f : A \to B$. In other words, if we can construct a bijective function $f : A \to B$, then $|A| = |B|$.

Example 34.1. Consider these two sets:

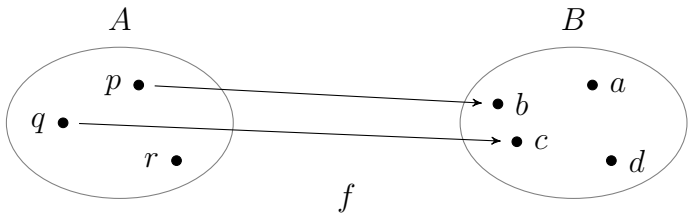
Are these two sets the same size? We will know that they are the same size if we can construct a bijective function between them. Let's try. Let's build an injective function f from A to B:



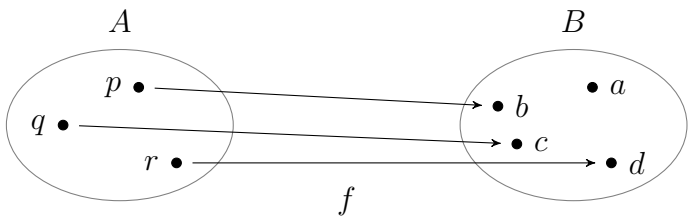
First, let's pair off p and b:



Next, let's pair off q and c:



Finally, let's pair off r and d:



Remark 34.6. Recall from Chapter 16 that an **injective function** maps each element from the domain to a *distinct* element in the codomain. It keeps the paths distinct, so to speak, in that each arrow goes to a different place. No two arrows end up at the same place.

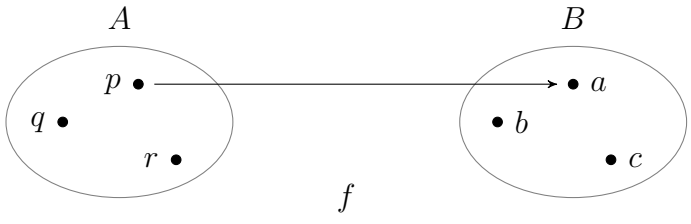
Remark 34.7. Can this function be **surjective**? Recall that a surjective function will cover all of the points in the codomain (every point in B will have an arrow pointing to it).

At this point, we have no more unpaired elements in A, so our function is completed.

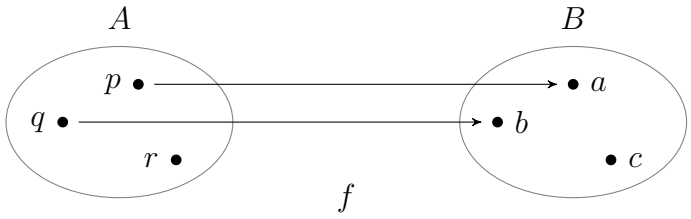
Nevertheless, we haven't covered all of the elements in B. One is untouched (namely, a). This is like a classroom with four students and only three desks. We put a student in each desk, but there's one student left over with no place to sit. This tells us that B is a bigger set than A.

Example 34.2. As another example, consider these two sets:

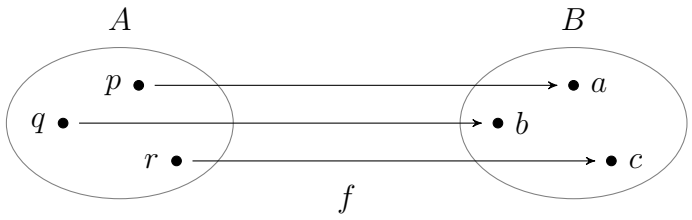
Can we construct a bijective function from A to B? Let's try. First, let's pair up p and a:



Next, let's pair up q and b:



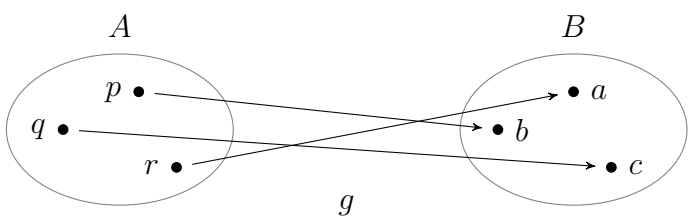
Finally, let's pair up r and c:



At this point, we have no more unpaired elements in A, so our function is completed.

Have we constructed a bijective function? Yes, we have. There are no extra items in B that haven't been paired off. Every single item in B has been uniquely paired up with an element in A.

Notice that f is not the only possible bijective function that we could construct here. There are others. For instance:



It doesn't really matter which bijective function we construct. All that matters is whether or not we can construct *one* of them.

If we *can* construct one (it doesn't matter which), then that tells us the two sets have exactly the same size, since by definition, a bijective function is essentially nothing more than an exact, one-to-one pairing up of the elements in two sets.

Remark 34.8. We have constructed a perfect, one-to-one pairing up of the items in these two sets. And since we can do this, we can conclude that the two sets have exactly the same number of elements. They are **equinumerous**.

By comparing examples Example 34.1 and Example 34.2, we can see that we will only be able to construct a bijective

function from one set to another if they are equinumerous. If one set contains more items than the other, then we will not be able to build a bijective function between them.

This is why building a bijective function between the two sets is enough to let us conclude that the two sets are the same size. Indeed, the bijective function explicates exactly what it means for the two sets to *be* the same size.

Remark 34.9. By definition, a bijective function puts two sets in exact, one-to-one alignment, with no leftovers. And that is precisely what it means for two sets to be the same size.

34.6 Summary

IN THIS CHAPTER, we looked at how to compare the sizes of two sets, without actually counting the number of objects contained in them.

- If we **count** the items in a set to determine its size, that works only for **finite** sets.
- **Infinity** is not a particular number (an element in $\mathbb{N}$). Rather it is the **size** of a never-ending sequence like $\mathbb{Z}^+$ or $\mathbb{N}$.
- We can **compare** sets for size, and we don't need to count them.
- If we can construct a **bijective** function (i.e., an **isomorphism**) between two sets, then we know they must be **equinumerous**.

[35]

LISTING INFINITIES

WE CAN COMPARE TWO SETS for size, without having to count them. As we saw in Chapter 34, if we can construct an **isomorphism** between the sets A and B, then it follows that A and B are the **same size**.

We can use this technique on **infinite** sets too. If we can construct an isomorphism between two infinite sets A and B, then we know that A and B must be the same size.

35.1 A Canonical Infinite Set

WHEN YOU MEASURE THINGS, you usually have to pick something as the "gold standard," i.e., the **canonical** (official) version that you compare everything else to.

In our case, we need to pick some infinite set, and agree that we will treat it as our canonical example. Once we have agreed that this particular set is the "gold standard" against which all others will be measured, then we will be able to compare other infinite sets to it.

Terminology. We need to elect one particular infinite set to serve as the standard that we will compare other infinite sets against. Here, we will elect the natural numbers for this role: $\mathbb{N}$ will be the **canonical** example of an infinite set.

Which set should we choose as our canonical example? There are options, but the natural numbers is a natural and simple choice. So, let's take the natural numbers $\mathbb{N}$ as our standard example of an infinite set. Let's agree that $\mathbb{N}$ will be the canonical version of an infinite set as we proceed.

Definition 35.1 (The canonical infinite set)**.** We will say that the natural numbers $\mathbb{N}$ is the **canonical infinite set**.

Now that we have designated $\mathbb{N}$ as our canonical infinite set, let's designate a special symbol to indicate its size. In popular culture, I think a lot of people know this symbol:

$$\infty$$

Remark 35.1. The infinity symbol ("∞," i.e., a sideways figure eight) is also called a **lemniscate**, and it is basically a picture of a ribbon that you could go round and round without end. It sometimes has other uses in math, but it is not used to indicate the *size* of infinite sets. For that, mathematicians use the Hebrew letter **Aleph** ($\aleph$). This symbol was first used by Georg Cantor (1845–1918), who first worked out in rigorous detail the material we are covering here.

But mathematicians have chosen a different symbol to denote the size of $\mathbb{N}$. It is this:

$$\aleph_0$$

You can pronounce that out loud as "Aleph-zero," or "Aleph-null," or "Aleph-nought," or something along those lines.

Instead of ∞, let's use Aleph-zero to denote the size of $\mathbb{N}$. Hence, if we want to write that the *size* (i.e., the **cardinality**) of the natural numbers $\mathbb{N}$ is $\aleph_0$, we can write this:

$$|\mathbb{N}| = \aleph_0$$

Read that out loud like this: "the cardinality (or size) of the set of natural numbers is Aleph-zero."

Definition 35.2 (The size of $\mathbb{N}$)**.** We will denote the **cardinality** (size) of the natural numbers $\mathbb{N}$ with the **Aleph-zero** symbol, namely $\aleph_0$. In other words: $|\mathbb{N}| = \aleph_0$.

This is not a **particular number**, like "10" or "$808,234,093$" are. Rather, it is a **size**. It designates the *size* of $\mathbb{N}$. We need a special symbol like this because $\mathbb{N}$ extends outwards, number after number, forever. We can't use a number to name that size, so we have to use a special, invented symbol ($\aleph_0$).

Remark 35.2. Recall from Chapter 34 that infinity is not a particular number, i.e., it is not an element in the natural numbers. Hence, when we say that $\aleph_0$ is not a particular number, we mean that it does not denote any particular element of $\mathbb{N}$. It is just a special symbol to designate the **size** of $\mathbb{N}$.

35.2 Comparing Infinities

WITH $\mathbb{N}$ AS OUR CANONICAL EXAMPLE of an infinite set, we can now compare other infinite sets to $\mathbb{N}$, to see if those other sets are the same size or not. And as we said, we will do this by constructing an isomorphism between them. If we can build an isomorphism from $\mathbb{N}$ to the others, then we can conclude that they are the same size (they *also* have the size $\aleph_0$).

Example 35.1. As a first example, let's take the set of integers $\mathbb{Z}$, from 0 on up. So, we're taking 0, and then all of the positive integers, and we're putting those into their own set. Let's call this set $\mathbb{Z}^{0+}$. It is this:

$$\mathbb{Z}^{0+} = \{0, 1, 2, 3, \dots\}$$

Let's compare this to our canonical example $\mathbb{N}$, and let's see if we can build an isomorphism. If we can do that, then we can conclude that $\mathbb{N}$ and $\mathbb{Z}^{0+}$ are the same size.

How can we build an isomorphism from $\mathbb{N}$ to $\mathbb{Z}^{0+}$? In this case, it's easy to see that these two sets line up exactly, so we can just map $0 \mapsto 0$, $1 \mapsto 1$, $2 \mapsto 2$ and so on. Let's draw the mapping sideways, like this:

$$\begin{array}{rccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & \dots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{Z}^{0+}: & 0 & 1 & 2 & 3 & \dots \end{array}$$

This mapping from $\mathbb{N}$ to $\mathbb{Z}^{0+}$ is clearly a bijective function, and so it is an **isomorphism** between these two sets. Hence, we can conclude that $\mathbb{Z}^{0+}$ is the **same size** as $\mathbb{N}$. And since $\mathbb{N}$ has a size of $\aleph_0$, then we can conclude that $\mathbb{Z}^{0+}$ is also that big. It too has a size of $\aleph_0$. Hence:

$$|\mathbb{Z}^{0+}| = \aleph_0$$

Example 35.2. Now, let's take the zero and all of the *negative* numbers from $\mathbb{Z}$, and let's put them into a set all their own. Let's call this set $\mathbb{Z}^{0-}$. It is this:

$$\mathbb{Z}^{0-} = \{\dots, -3, -2, -1, 0\}$$

Is this the same size as $\mathbb{N}$? If we can construct an isomorphism from $\mathbb{N}$ to $\mathbb{Z}^{0-}$, then we can conclude that it is the same size.

So, can we construct an isomorphism? Yes, by just listing the negative numbers in reverse order. Like this:

$$\begin{array}{rccccc} \mathbb{N} & 0 & 1 & 2 & 3 & \dots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{Z}^{0-} & 0 & -1 & -2 & -3 & \dots \end{array}$$

Since we can construct an isomorphism here, we can conclude that $\mathbb{Z}^{0-}$ is the same size as $\mathbb{N}$, and hence it has the size $\aleph_0$ too:

Remark 35.3. To be more explicit, the function looks like this:

$$\begin{array}{lll} f(0) = 0 & f(1) = 1 & f(2) = 2 \\ f(3) = 3 & f(4) = 4 & \dots \end{array}$$

Remark 35.4. To be more explicit, the function looks like this:

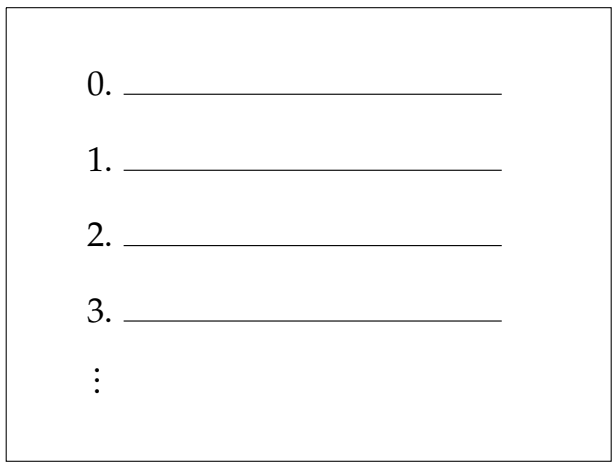
$$\begin{array}{lll} f(0) = 0 & f(1) = -1 & f(2) = -2 \\ f(3) = -3 & f(4) = -4 & \dots \end{array}$$

$$|\mathbb{Z}^{0-}| = \aleph_0$$

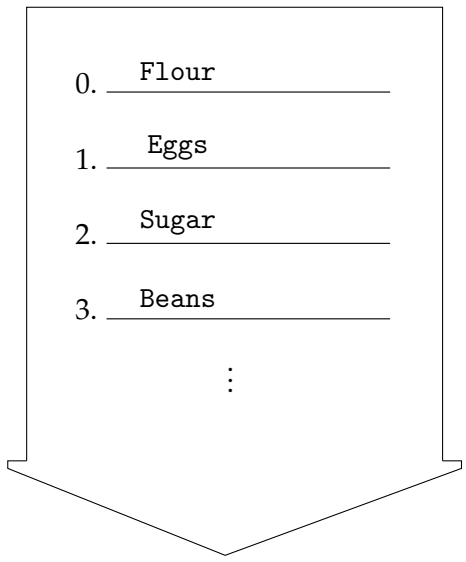
35.3 Listable Sets

THERE IS ANOTHER WAY to think about what we have been doing here. When we construct an isomorphism from $\mathbb{N}$ to some other set A, what exactly are we doing?

Well, one way to look at it is this: we are **listing** the elements in A, one by one. Imagine that we have an empty shopping list which is infinitely long. Something like this:



Remark 35.5. Think of this as an infinitely long shopping list. E.g.,



To construct an isomorphism from $\mathbb{N}$ to any set A, we basically need to fill in each blank on this list with an item from A. And to do that, we just write in the items from A, one by one.

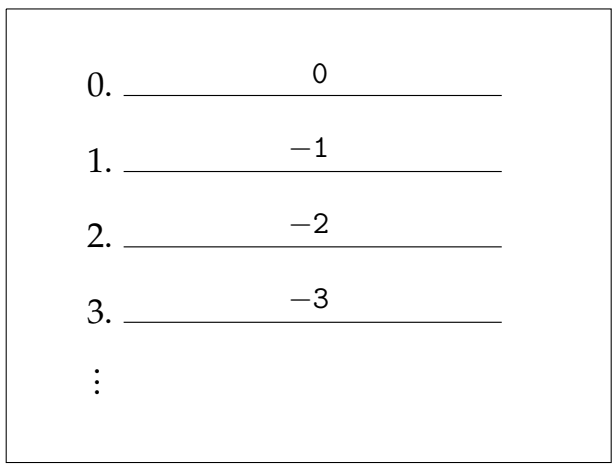
For instance, with $\mathbb{Z}^{0-}$, first we take "0" and we write it in the first blank:

0. 0
1. _____
2. _____
3. _____
⋮

Then we take "-1" and we write it in the next blank:

0. 0
1. -1
2. _____
3. _____
⋮

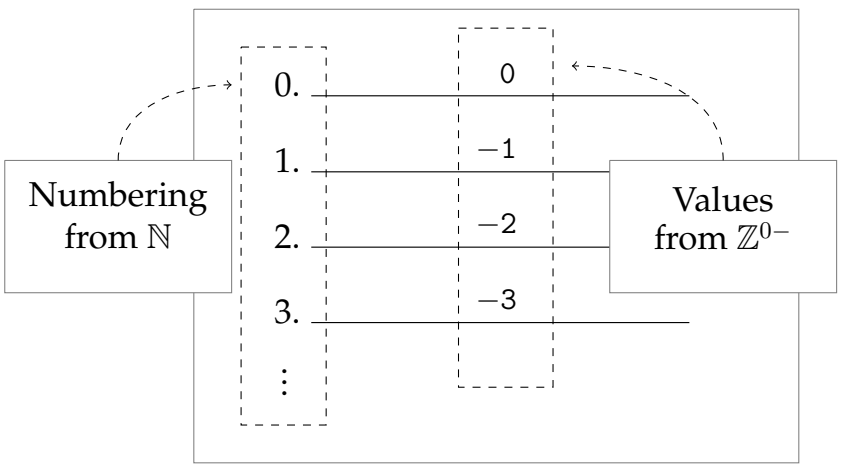
Then we write "-2" in the next blank, then "-3" in the blank after that, and so on:



Of course, for infinite sets, we humans could never complete the process of **listing out** all of the elements in $\mathbb{Z}^{0-}$, since that would go on and on, forever. Nevertheless, we can easily imagine filling it in, endlessly.

So, we can see an isomorphism from $\mathbb{N}$ to another set A as a **listing** of all the elements in A. The set $\mathbb{N}$ really just provides the numbering of the blanks that we see going down the left side:

Ponder. If you like, you can imagine for the sake of the argument that there is something unlimitedly intelligent and powerful (such as the being that many call "God") which has the computational capacity to be able to complete the entire list here, all at once.



So, let us say that an isomorphism from $\mathbb{N}$ to a set A is a **listing** (or synonymously, an **enumeration**) of A, since the

isomorphism "lists out" or enumerates all the items in A, by pairing each one up with a number from $\mathbb{N}$.

Terminology. A **listing** (or **enumeration**) of an infinite set A is an isomorphism f from $\mathbb{N}$ to A. In essence, f gives us a way to list every element in A.

Definition 35.3 (Listings)**.** For any infinite set A, we will say that an isomorphism $f : \mathbb{N} \to A$ is a **listing** (or synonymously, an **enumeration**) of A.

If we can construct a listing of an infinite set A, then let us say that it is **listable** (or synonymously, **enumerable**). So, if we have an infinite set A, and we are able to construct a listing of it (i.e., an isomorphism of it), then we can say that it is listable. It's items can be listed or enumerated on this infinitely long shopping list that we have been talking about.

Terminology. A set A is **listable** (or **enumerable**) if a listing (enumeration) can be constructed. If you prefer to imagine a being with unlimited computational power (e.g., such as the being many call "God"), then you might imagine that a set is listable if it is the sort of thing that God could list out or enumerate on an infinitely long shopping list.

Definition 35.4 (Listability)**.** For any infinite set A, we will say that A is **listable** (or synonymously, **enumerable**) if a listing (enumeration) can be constructed.

Why are we setting down these definitions? The reason is this. It turns out that not every infinite set is listable. There are certain infinite sets for which no listing can be constructed. That is a topic we will turn to soon.

35.4 Summary

IN THIS CHAPTER, we looked at listing out or enumerating the items in a set.

- We choose the **natural numbers** $\mathbb{N}$ as the **canonical** example of an infinite set. We designate a special symbol

for the **size** of this set: $\aleph_0$ (pronounced "Aleph-zero," or "Aleph-null," or "Aleph-nought").

- A **listing** (or synonymously, an **enumeration**) of a set A is an **isomorphism** f from $\mathbb{N}$ to A.
- A set is **listable** (or synonymously, **enumerable**) if a **listing** of it can be constructed.

[36]

LISTABLE INFINITIES

AS WE SAW IN CHAPTER 35, an infinite set A is **listable** (or synonymously, **enumerable**) if we can construct an **isomorphism** f from $\mathbb{N}$ to A. In effect, an isomorphism shows us how to list out each element of A.

So which infinite sets are listable? In this chapter, we will look at a few different examples of listable infinities.

Ponder. What are some infinite sets that you think might be listable? Do you think any of the sets of numbers that we discussed in Part 6 are listable? Do you think any of them might *not* be listable?

36.1 Subsets of $\mathbb{N}$

A SURPRISING THING about the natural numbers is that certain subsets of the natural numbers are just as large as the entire set of natural numbers. This is rather counter intuitive. Let’s look at some examples.

Remark 36.1. We normally think that a subset of a set is a *smaller* slice of that set, so how can a subset of $\mathbb{N}$ be just as large as the whole of $\mathbb{N}$? With infinite sets, our intuitions mislead us.

Example 36.1. Think about the natural numbers, i.e., $\mathbb{N}$:

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$$

Now consider just the even ones. Let's call this set $\mathbb{N}^E$:

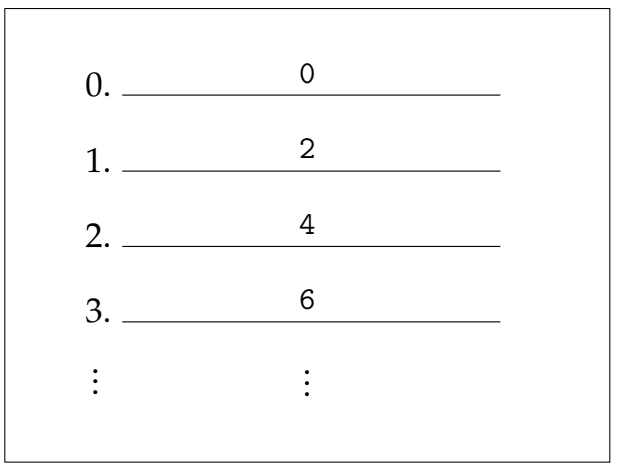
$$\mathbb{N}^E = \{0, 2, 4, 6, \ldots\}$$

Both of these sets are infinite, since each of them goes on forever. But here is the question: are they the same size?

Intuitively, it might seem like these two sets should be different in size. $\mathbb{N}^E$ has only even numbers in it, while $\mathbb{N}$ has even *and* the odd numbers in it. Shouldn't $\mathbb{N}^E$ therefore be *half* the size of $\mathbb{N}$?

Well, let's construct a listing. Here it is:

Remark 36.2. How do we determine if two sets are the same size? We use the trick from Chapter 34 and Chapter 35: if we can construct a **listing** (an **isomorphism**), then the two must be the same size. Can we do this?



Or, to write it out horizontally:

$$\begin{array}{lcccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & \ldots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{N}^E: & 0 & 2 & 4 & 6 & \ldots \end{array}$$

Remark 36.3. It is clear that this listing is indeed an **isomorphism**. It is a reversible, bijective function from $\mathbb{N}$ to $\mathbb{N}^E$.

You can see that we can in fact build up a listing like this. The trick lies in the fact that both $\mathbb{N}$ and $\mathbb{N}^E$ are endless, so we

will never run out of numbers. For each number from $\mathbb{N}$, we take the next even number from $\mathbb{N}^E$, and we just repeat that again and again, forever.

What can we conclude? We must conclude that $\mathbb{N}^E$ is not in fact half the size of $\mathbb{N}$. On the contrary, $\mathbb{N}^E$ and $\mathbb{N}$ are the same size! They are both of size $\aleph_0$:

$$|\mathbb{N}| = \aleph_0 \qquad \text{and} \qquad |\mathbb{N}^E| = \aleph_0$$

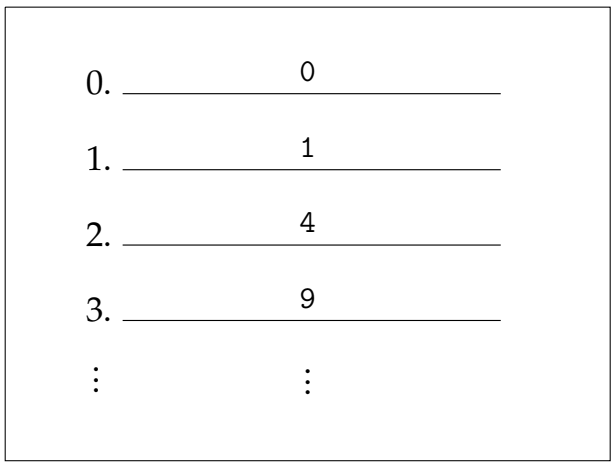
Remark 36.4. Here is where our intuitions mislead us. We initially think that $\mathbb{N}^E$ must be half the size of $\mathbb{N}$, but that's only because we forget that both of these sets are endless sequences. Once we remember that neither one has an end, it is easier to see (by the listing) that they are the same size.

Example 36.2. Consider the square of every natural number. That is, take each natural number n, and square it: n^2. For instance, 0^2 is 0, 1^2 is 1, 2^2 is 4, 3^2 is 9, 4^2 is 16, and so on. Let's put all of these into a set, and call it $\mathbb{N}^{n^2}$. Like this:

$$\mathbb{N}^{n^2} = \{0, 1, 4, 9, 16, \ldots\}$$

Is this set the same size as $\mathbb{N}$? Yes it is, since we can list it:

Remark 36.5. The square of a number is that number times itself. So, 2^2 is just 2 times 2 (i.e., 4), 3^2 is just 3 times 3 (i.e., 9), 4^2 is just 4 times 4 (i.e., 16), and so on.



Or, to write it out horizontally:

$$\begin{array}{rccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & \ldots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{N}^E: & 0 & 1 & 4 & 9 & \ldots \end{array}$$

Hence, $\mathbb{N}^{n^2}$ and $\mathbb{N}$ are the same size too. They are both of size $\aleph_0$:

$$|\mathbb{N}| = \aleph_0 \qquad \text{and} \qquad |\mathbb{N}^{n^2}| = \aleph_0$$

36.2 How Many Integers?

LET'S TURN OUR ATTENTION to the integers. As we know, the integers include all the natural numbers, plus all the negative whole numbers too:

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

Ponder. The natural numbers $\mathbb{N}$ appear to have a starting point (namely, 0), and then it appears to increase forever from there: after 0 there's 1, then 2, and so on. The integers $\mathbb{Z}$ appear to be a little different in this regard, in that it appears that they *don't* have a starting point. Zero looks like it's in the middle, and then it appears to extend outwards from there in both directions endlessly. But do you think this is really true? Remember: these are just sets.

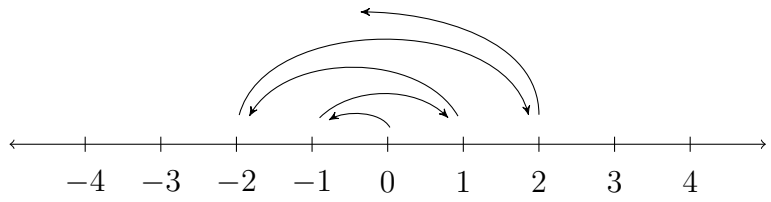
Intuitively, it might seem like $\mathbb{Z}$ should be *twice* the size of $\mathbb{N}$, since it includes all the same numbers as $\mathbb{N}$, *plus* a negative version of each number too. But is it really the case that $\mathbb{Z}$ is twice as big as $\mathbb{N}$?

Again, we can use our listing trick to find out. If $\mathbb{Z}$ is the same size as $\mathbb{N}$, then we will be able to construct a **listing** (**isomorphism**) of $\mathbb{Z}$. So, is such a listing possible?

One of the things that is required to make a listing is a definite starting point. We have to be able to start *somewhere*, and then there has to be a repeatable way to find the *next* number to put down.

Well, what is the "starting" point for listing out $\mathbb{Z}$? Where is the "smallest number"? There isn't a smallest one, because no matter how far back in the negatives we go, there are always more beyond that. So, it might appear that we cannot in fact **list** all of $\mathbb{Z}$, because we just can't find a *starting point*.

However, there is a trick. Let's start in the middle (at 0), and then work our way outwards. We'll follow this pattern:



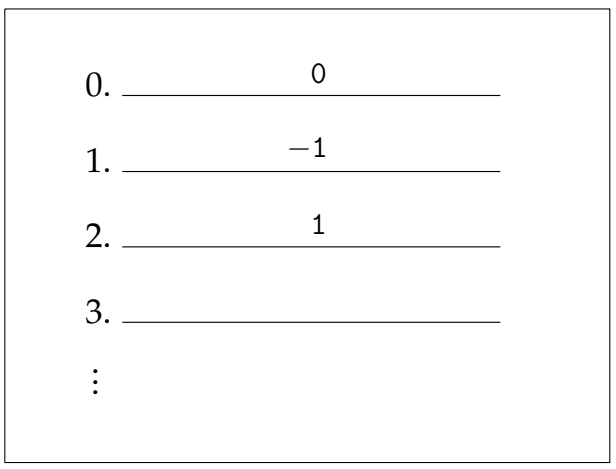
So, to make this listing, we first write down 0:

0. 0
1.
2.
3.
⋮

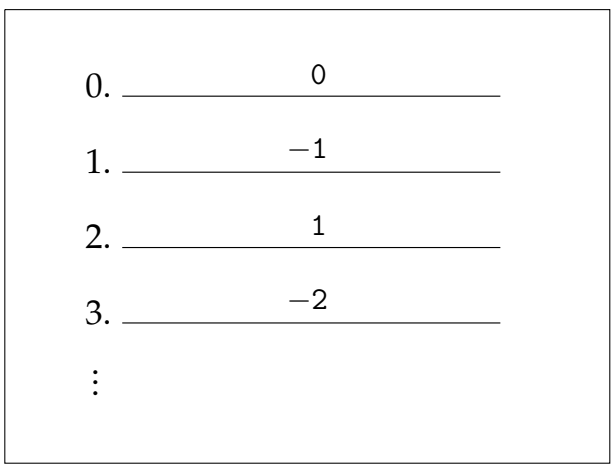
Then we write down -1:

0. 0
1. -1
2.
3.
⋮

Then we write down 1:



Then we move to -2:



Then we write down 2, then -3, then 3, and so on. Here is the listing, written horizontally:

$$\begin{array}{lcccccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & 4 & 5 & \dots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{Z}: & 0 & -1 & 1 & -2 & 2 & -3 & \dots \end{array}$$

We just start with 0, and then we write the negative and the positive version of each integer after that. Basically, we

just rearrange the integers, so that they start at 0 and then proceed only to the right, like this:

$$\mathbb{Z}: \quad 0 \quad -1 \quad 1 \quad -2 \quad 2 \quad -3 \quad 3 \quad -4 \quad 4 \quad \dots$$

By doing this, we then have a **listing** of the integers that has a definite starting place, and a way to proceed forward, one negative and positive integer at a time, forever.

Hence, the integers $\mathbb{Z}$ are listable too, and hence they also are of size $\aleph_0$:

$$|\mathbb{N}| = \aleph_0 \qquad \text{and} \qquad |\mathbb{Z}| = \aleph_0$$

Remark 36.6. Again, our intuitions have simply misled us. While it might seem at first that $\mathbb{Z}$ should be a bigger set than $\mathbb{N}$, by using our listing trick, we can see that there is a precise one-to-one pairing of $\mathbb{N}$ and $\mathbb{Z}$ (which is only possible because both sets are endless).

36.3 How Many Rational Numbers?

WHAT ABOUT THE RATIONALS $\mathbb{Q}$, i.e., the fractions? Is the size of $\mathbb{Q}$ the same as $\mathbb{N}$ and $\mathbb{Z}$? On the face of it, it might seem that there are *way more* fractions than there are natural numbers or integers. After all, between 0 and 1 there are *infinitely many* fractions! So surely $\mathbb{Q}$ is *much* bigger. Or do our intuitions mislead us here too?

In order to determine the answer to this, we can use the same technique we have been using: we need to see if there is a way to list $\mathbb{Q}$. If we can find a listing (isomorphism) of $\mathbb{Q}$, then we will know that $\mathbb{Q}$ is the same size as $\mathbb{N}$. Is it possible to find such a listing?

There are very many fractions. Let's write some of them out. First, let's start by listing out fractions that can be formed with a "1" as the bottom number. Like this:

$$\frac{1}{1} \quad \frac{2}{1} \quad \frac{3}{1} \quad \frac{4}{1} \quad \dots$$

We also have "$\frac{0}{1}$", so let's push that one on to the stack, up in the front (on the left side):

Remark 36.7. These are just the positive whole numbers, written as fractions. To write "1" as a fraction, we write "$\frac{1}{1}$," to write "2" as a fraction we write "$\frac{2}{1}$," for "3" we write "$\frac{3}{1}$," and so on. And "$\frac{0}{1}$" is just "0."

$$\frac{0}{1} \quad \frac{1}{1} \quad \frac{2}{1} \quad \frac{3}{1} \quad \frac{4}{1} \quad \dots$$

We also have negative fractions, so let's add those in too:

$$\dots \quad -\frac{4}{1} \quad -\frac{3}{1} \quad -\frac{2}{1} \quad -\frac{1}{1} \quad \frac{0}{1} \quad \frac{1}{1} \quad \frac{2}{1} \quad \frac{3}{1} \quad \frac{4}{1} \quad \dots$$

Remark 36.8. Since these are the positive and negative whole numbers (written as fractions), this much of our list is equivalent to (or better: isomorphic to) the integers $\mathbb{Z}$. Can you see how to build an isomorphism between this and $\mathbb{Z}$?

At this point, we have a list (extending forever in both directions) that is comprised of all fractions that can be formed with a "1" as the bottom number.

Next, let's think about all fractions that can be formed with a "2" as the bottom number. We can follow the same pattern we used for fractions with a "1" as the bottom number:

$$\dots \quad -\frac{4}{2} \quad -\frac{3}{2} \quad -\frac{2}{2} \quad -\frac{1}{2} \quad \frac{0}{2} \quad \frac{1}{2} \quad \frac{2}{2} \quad \frac{3}{2} \quad \frac{4}{2} \quad \dots$$

Remark 36.9. Some of these fractions can be simplified to others. For instance, $\frac{2}{2} = \frac{1}{1}$, and $\frac{4}{2} = \frac{2}{1}$. But our goal here isn't to write out only fully simplified fractions. Our goal is simply to systematically write out all possible fractions of the form $\frac{m}{n}$.

Now we have a list (extending forever in both directions) that is comprised of all fractions that can be formed with a "2" as the bottom number. Let's combine that with the fractions with a "1" for the bottom number. Like this:

$$\dots \quad -\frac{4}{1} \quad -\frac{3}{1} \quad -\frac{2}{1} \quad -\frac{1}{1} \quad \frac{0}{1} \quad \frac{1}{1} \quad \frac{2}{1} \quad \frac{3}{1} \quad \frac{4}{1} \quad \dots$$

$$\dots \quad -\frac{4}{2} \quad -\frac{3}{2} \quad -\frac{2}{2} \quad -\frac{1}{2} \quad \frac{0}{2} \quad \frac{1}{2} \quad \frac{2}{2} \quad \frac{3}{2} \quad \frac{4}{2} \quad \dots$$

We can next add all fractions that can be formed with a "3" as the bottom number:

$$\dots \quad -\frac{4}{1} \quad -\frac{3}{1} \quad -\frac{2}{1} \quad -\frac{1}{1} \quad \frac{0}{1} \quad \frac{1}{1} \quad \frac{2}{1} \quad \frac{3}{1} \quad \frac{4}{1} \quad \dots$$

$$\dots \quad -\frac{4}{2} \quad -\frac{3}{2} \quad -\frac{2}{2} \quad -\frac{1}{2} \quad \frac{0}{2} \quad \frac{1}{2} \quad \frac{2}{2} \quad \frac{3}{2} \quad \frac{4}{2} \quad \dots$$

$$\dots \quad -\frac{4}{3} \quad -\frac{3}{3} \quad -\frac{2}{3} \quad -\frac{1}{3} \quad \frac{0}{3} \quad \frac{1}{3} \quad \frac{2}{3} \quad \frac{3}{3} \quad \frac{4}{3} \quad \dots$$

And then the same for "4" as the bottom number, "5," and so on, forever:

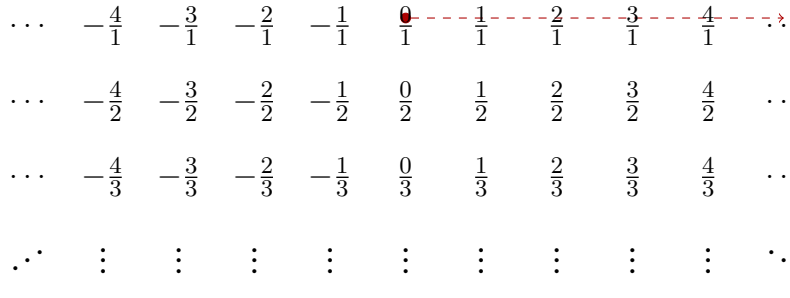
$$\begin{array}{ccccccccccc}
\cdots & -\frac{4}{1} & -\frac{3}{1} & -\frac{2}{1} & -\frac{1}{1} & \frac{0}{1} & \frac{1}{1} & \frac{2}{1} & \frac{3}{1} & \frac{4}{1} & \cdots \\
\cdots & -\frac{4}{2} & -\frac{3}{2} & -\frac{2}{2} & -\frac{1}{2} & \frac{0}{2} & \frac{1}{2} & \frac{2}{2} & \frac{3}{2} & \frac{4}{2} & \cdots \\
\cdots & -\frac{4}{3} & -\frac{3}{3} & -\frac{2}{3} & -\frac{1}{3} & \frac{0}{3} & \frac{1}{3} & \frac{2}{3} & \frac{3}{3} & \frac{4}{3} & \cdots \\
\iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

At this point, we have some idea of what is involved in writing out all possible fractions with the form $\frac{m}{n}$. Let's turn now to the question of whether there is a way to make a listing of them (i.e., an isomorphism with $\mathbb{N}$).

Let's imagine putting our pen down on one fraction, and then systematically drawing a path that will take us through each fraction. Can we find such a path?

Remark 36.10. If we can draw such a path with a pen, then that is essentially a listing of all of these fractions (we can just write down each fraction as we traverse over it with our pen).

In this picture, there is no straight-lined path that will work. If we start at "$\frac{0}{1}$" and move to the right, we will never get to the end of that row, and never get back to another row:



Similarly, if we start at "$\frac{0}{1}$" and move to the left, we will get stuck going left forever, because that row never ends either:

$$\begin{array}{cccccccccccc}
\cdots & -\frac{4}{1} & -\frac{3}{1} & -\frac{2}{1} & -\frac{1}{1} & \frac{0}{1} & \frac{1}{1} & \frac{2}{1} & \frac{3}{1} & \frac{4}{1} & \cdots \\
\cdots & -\frac{4}{2} & -\frac{3}{2} & -\frac{2}{2} & -\frac{1}{2} & \frac{0}{2} & \frac{1}{2} & \frac{2}{2} & \frac{3}{2} & \frac{4}{2} & \cdots \\
\cdots & -\frac{4}{3} & -\frac{3}{3} & -\frac{2}{3} & -\frac{1}{3} & \frac{0}{3} & \frac{1}{3} & \frac{2}{3} & \frac{3}{3} & \frac{4}{3} & \cdots \\
\iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

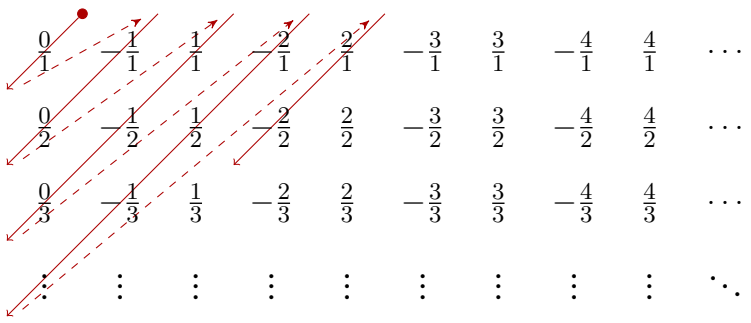
For this first row, let’s use the re-arranging trick that we used for $\mathbb{Z}$: let’s write the negative and then positive versions of each fraction in a sequence. The first row becomes this:

$$\frac{0}{1} \quad -\frac{1}{1} \quad \frac{1}{1} \quad -\frac{2}{1} \quad \frac{2}{1} \quad -\frac{3}{1} \quad \frac{3}{1} \quad -\frac{4}{1} \quad \frac{4}{1} \quad \cdots$$

Now we can start on the left (at $\frac{0}{1}$), and move only to the right. Let’s do the same re-arrangement for all of the rows:

$$\begin{array}{cccccccccc}
\frac{0}{1} & -\frac{1}{1} & \frac{1}{1} & -\frac{2}{1} & \frac{2}{1} & -\frac{3}{1} & \frac{3}{1} & -\frac{4}{1} & \frac{4}{1} & \cdots \\
\frac{0}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{2}{2} & \frac{2}{2} & -\frac{3}{2} & \frac{3}{2} & -\frac{4}{2} & \frac{4}{2} & \cdots \\
\frac{0}{3} & -\frac{1}{3} & \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} & -\frac{3}{3} & \frac{3}{3} & -\frac{4}{3} & \frac{4}{3} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Now our grid of fractions extends only to the right and downwards (not to the left). Can we draw a path now? Is there a way to draw a path through these fractions, without getting stuck going down a path that never ends? There is, if we draw a diagonal zigzag. Like this:



We start at "$\frac{0}{1}$," then we go up to "$-\frac{1}{1}$" and do a diagonal stroke to the left until we get to the left edge (at "$\frac{0}{2}$"). Then we go up to "$\frac{1}{1}$" and do a diagonal stroke again (down to "$\frac{0}{3}$"), then we go up again, and diagonal down again, and so on.

Tthis particular way of traversing the nodes will never get stuck. We will just keep going on forever, in the correct direction. Hence, this gives us a way to list the fractions. If we write down each fraction as we traverse over it with our pen, we get this:

$$\mathbb{Q}: \quad \frac{0}{1} \quad -\frac{1}{1} \quad \frac{0}{2} \quad \frac{1}{1} \quad -\frac{1}{2} \quad \frac{0}{3} \quad -\frac{2}{1} \quad \frac{1}{2} \quad -\frac{1}{3} \quad \cdots$$

Remark 36.11. There are other ways to traverse these fractions with a pen so that we never get stuck. Can you see any more? There is even a way to do it without rearranging each row.

And we can use this to make a listing:

0. $^0/_1$
1. $^{-1}/_1$
2. $^0/_2$
3. $^1/_1$
⋮ ⋮

Or, if you want to write it horizontally:

$$\begin{array}{lccccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & 4 & 5 & \dots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{Z}: & \frac{0}{0} & -\frac{1}{1} & \frac{0}{2} & \frac{1}{1} & -\frac{1}{2} & \frac{0}{3} & \dots \end{array}$$

So, it turns out that we *can* construct a **listing** (isomorphism) of the rational numbers after all. Hence, $\mathbb{Q}$ and $\mathbb{N}$ are the same size, along with $\mathbb{Z}$:

$$|\mathbb{N}| = \aleph_0 \qquad \text{and} \qquad |\mathbb{Z}| = \aleph_0 \qquad \text{and} \qquad |\mathbb{Q}| = \aleph_0$$

All three of these sets of numbers have the same size: $\aleph_0$. There are just as many fractions as there are integers, and there are just as many integers as there are natural numbers.

Remark 36.12. Again, our intuitions can mislead us. For although our intuitions might make us think that surely $\mathbb{Q}$ is a bigger set than $\mathbb{N}$ and $\mathbb{Z}$, once we sit down and construct a listing, we can see that even $\mathbb{Q}$ is the same size as $\mathbb{N}$.

36.4 Summary

IN THIS CHAPTER, we looked at different examples of infinite, listable sets. Surprisingly, all of them turn out to be the same size as the natural numbers $\mathbb{N}$. In each case, this is revealed by finding a way to list them, one by one.

- Infinite subsets of $\mathbb{N}$ (like just the even numbers) turn out to have a size of $\aleph_0$, just like $\mathbb{N}$, even though our intuitions might suggest that subsets of a set should be smaller than the parent set.
- $\mathbb{Z}$ have a size of $\aleph_0$, just like $\mathbb{N}$, even though our intuitions might suggest that $\mathbb{Z}$ should be twice as large as $\mathbb{N}$ because $\mathbb{Z}$ has negative numbers in addition to positive numbers.
- $\mathbb{Q}$ have a size of $\aleph_0$ too, just like $\mathbb{N}$, even though there are infinitely many fractions between every natural number.

[37]

UNLISTABLE INFINITIES

CAN WE LIST THE REAL NUMBERS $\mathbb{R}$? It turns out that we cannot. The real numbers $\mathbb{R}$ are unlistable. So, here we encounter an infinite set that is *not* listable.

Now, how could we possibly know that $\mathbb{R}$ is unlistable? In Chapter 36, we deduced that the integers $\mathbb{Z}$ and the rational numbers $\mathbb{Q}$ are the same size as the natural numbers $\mathbb{N}$. How did we know this? We figured this out by finding an isomorphism between them. In other words, we found a way to list all of $\mathbb{Z}$, and we found a way to list all of $\mathbb{Q}$.

However, we had to do a little work to find a way to list $\mathbb{Q}$. The listing strategy for $\mathbb{Q}$ was not entirely obvious at first. Nevertheless, after some thinking, we did find a way to list all the fractions in $\mathbb{Q}$, proving that $\mathbb{Q}$ and $\mathbb{N}$ are the same size.

When it comes to the real numbers $\mathbb{R}$, who's to say that there isn't some clever way to list them all out, and humans

Ponder. Can you think of a way to *prove* that an infinite set is not listable? What sorts of strategies might you try?

just haven't found it yet? How can we say with certainty that the real numbers are definitely *not* listable?

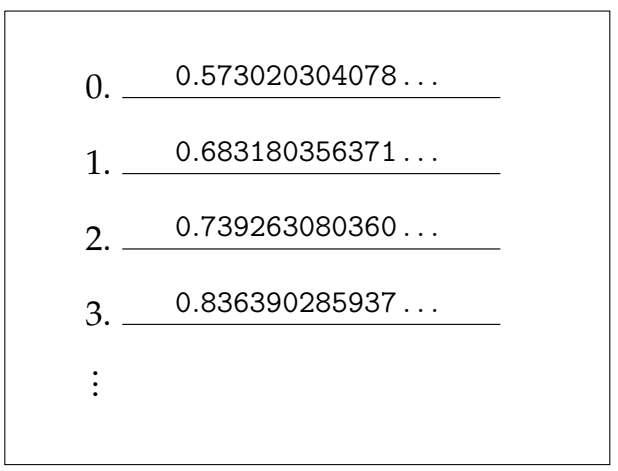
The answer is that we can prove it. How do we prove it? We do it by finding real numbers that cannot be included in any listing of $\mathbb{R}$. There is a particular trick to this, which we call **diagonalization**, and that is what we will look at here.

37.1 The Diagonalization Trick

LET'S SUPPOSE THAT WE DO HAVE A LISTING of all of the real numbers. As we said, this is impossible, but let's pretend that we do have such a listing for the sake of the argument.

Real numbers have an infinite number of digits after decimal places, so such a listing would look something like this:

Remark 37.1. Note that in this picture, each number written down has infinitely many digits after the decimal point. So each number would keep going to the right, forever.



Let's make this a little more precise. Let's draw this same listing a little differently. Let's start by re-drawing the listing with a little more space around all of the digits, like this:

$$
\begin{array}{ll}
0) & 0.5730\ldots \\
1) & 0.6831\ldots \\
2) & 0.7392\ldots \\
3) & 0.8363\ldots \\
\vdots & \vdots
\end{array}
$$

On the left side, each line is numbered with a number from the natural numbers $\mathbb{N}$. Let's highlight these line numbers, and add a separator to make it clear that these line numbers are not part of the real numbers on the right:

$$
\begin{array}{l|l}
\mathbf{0)} & 0.5730\ldots \\
\mathbf{1)} & 0.6831\ldots \\
\mathbf{2)} & 0.7392\ldots \\
\mathbf{3)} & 0.8363\ldots \\
\vdots & \vdots
\end{array}
$$

Each real number on the right side of the separator has an infinite number of digits after the decimal point. Let's number each decimal position across the top, like this:

	0th		**1st**	**2nd**	**3rd**	**4th**	**...**
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

Now, at this point, we're going to use a little trick to construct a special real number. Let's call this number n. The way we perform this trick runs as follows.

We're going to build n one digit at a time. So let's make a slot for n near the top, and we'll fill it in as we go:

n	??	.	??	??	??	??	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

Next, we look at the digit on line 0 in the 0th position:

n	??	.	??	??	??	??	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

The digit there is "0." In our slot for n, let's fill in the 0th position with a digit that is *not* "0." Let's pick "1":

n	1	.	??	??	??	??	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

Remark 37.2. Note what we just did: we just made sure that our number n differs from the number listed on line 0 in its 0th digit. However many other digits that n and the number at line 0 might have in common, they will differ *at least* in this one digit. Hence, our number n will *not* be the same number as the number on line 0.

Next, let's move to line 1, and look at its 1st position digit:

n	1	.	??	??	??	??	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

It's "6." Let's now go back up to our slot for n, and let's fill in the 1st position of n with a digit that's *not* "6," say "4":

n	1	.	4	??	??	??	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

Remark 37.3. Notice again what we did: we made sure that our number n differs from the number that appears on line 1. No matter how many other digits n and the number on line 1 might have in common, they will differ at least in their 1st digit. Hence, so far, our number n is different from the number on line 0, and it is different from the number on line 1.

Next, let's turn to line 2, and note its 2nd position digit:

n	1	.	4	??	??	??	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

It's "3." For our n, let's fill in its 2nd position digit with something that's not "3," say "7":

n	1	.	4	7	??	??	...

	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

Remark 37.4. Again, note what we did. We made sure that our number n differs from the number on line 2. However many other digits n and the number on line 2 might have in common, they will differ at least in the digit in that 2nd position.

Let's now do the same for the 3rd digit on line 3. We can fill in n's 3rd position with any digit that is different from the one in the 3rd position of line 3. For instance:

n	1	.	4	7	2	??	...

	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

Remark 37.5. Now our number n is different from the number on line 3. However many other digits n and the number on line 3 have in common, they will differ at least in the digit in that 3rd position.

We then go on and on like this, filling in the 4th position of n with a digit that is different from the 4th digit of line 4, then likewise for the 5th position, and the 6th position, and so on.

Now, we obviously can't keep doing this ourselves, because these numbers are infinitely long and the list extends infinitely downwards. But it's easy enough to imagine how this would repeat. We have a simple procedure here that we can use to choose each subsequent digit to build n. It's this:

for the kth position of n, we just pick a digit that it different from the kth position on line k!

Let's write this down, as a kind of recipe for how to build the number n:

> To build the number n, fill in each digit as follows. To fill in the kth digit, fill it in with any digit x that is different from the digit in the kth position on the kth line.

We call this trick the **diagonalization** trick (or just "diagonalization"). We call it this because to carry it out, we go down the diagonal of the listing, and make sure that our number n differs at each point in the diagonal. Here are all the positions we went through, but highlighted so you can see how it's the diagonal:

Terminology. **Diagonalization** is the process of making sure that a sequence of items n differs from a list of other sequences by making it differ down the diagonal.

n	1	.	4	7	2	...	...
	0th		**1st**	**2nd**	**3rd**	**4th**	...
0)	0	.	5	7	3	0	...
1)	0	.	6	8	3	1	...
2)	0	.	7	3	9	2	...
3)	0	.	8	3	6	3	...
⋮	⋮		⋮	⋮	⋮	⋮	

37.2 An Unlistable Number

THE NUMBER WE JUST CONSTRUCTED (namely 1.472...) just *cannot* be listed in our listing. Why not? The reason is that we constructed it in such a way that it differs from every other number on the list. We built it to *be* different. Think about it:

- n is different from the number listed on line 0, because they differ in their 0th position.
- n is different from the number listed on line 1, because they differ in their 1st position.
- n is different from the number listed on line 2, because they differ in their 2nd position.
- n is different from the number listed on line 3, because they differ in their 3rd position.
- And so on for every possible number that is listed! No matter how far down the list we go, n will be different. No matter which line k we check, n is different from the number listed on line k, because it differs from that number in the kth position.

So, our number n simply cannot be on the list. It is impossible for it to be on the list!

Remark 37.6. The **diagonalization** trick has allowed us to construct a real number which, somehow, cannot be on the list.

37.3 $\mathbb{R}$ Is Unlistable

WHAT CONCLUSION ARE WE TO DRAW FROM THIS? We have seen that there is at least one real number that cannot be listed, and that is the number n that we constructed with the diagonalization trick. So what does that mean?

It means that the real numbers are unlistable (they are not **enumarable**). We supposed initially that we had a list of all real numbers. But then we built a real number that can't be on the list. So, the real numbers must *not* be listable.

Remark 37.7. Recall from Chapter 4 that a proof by contradiction runs as follows: we assume the opposite of what we want to prove, then we show with pure logic that this leads to a contradictory state of affairs. This shows that our initial assumption is not possible, and therefore the opposite must be true.

To be more exact, we have just carried out a proof **by contradiction**. Here are the steps, spelled out explicitly:

- We start by assuming that there *is* a listing of all of the real numbers.

- Next, we use the diagonalization trick to construct a real number n that *cannot* be on our supposed list.
- This puts us in a state of contradiction. For we are now in a state where (1) *all* real numbers are listed (we assumed this to start), but yet (2) there is at least *one* that is not on the list, namely n.
- So, we are in an impossible state of affairs, and when that happens, we must go back and find the step in our reasoning where we went wrong.
- There is nothing wrong with the way we constructed n. It is a very simple procedure that even a computer could follow: for the kth digit of n, pick a digit that differs from the kth digit on the kth line of the supposed list.
- So, that leaves our original assumption: namely, our assumption that the real numbers *are* listable. This is the only thing left that could be wrong.
- So, that assumption must be false. The opposite must be true. It must in fact be the case that the real numbers cannot be listed in the first place.

Recall that $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$ are all the same size: $\aleph_0$, because they are all listable. By contrast, the real numbers $\mathbb{R}$ are not listable. There are too many of them. Hence, the real numbers are **bigger** than $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$. The real numbers are infinite too, but they make up an even bigger infinity.

Since the size of $\mathbb{R}$ is bigger than $\aleph_0$, mathematicians say that the size of $\mathbb{R}$ is $\aleph_1$, pronounced "Aleph one":

$$|\mathbb{R}| = \aleph_1$$

The idea that the real numbers $\mathbb{R}$ cannot be listed is an amazing fact. Our intuition makes it easy enough to imagine something like the list of real numbers that we imagined

Remark 37.8. $\mathbb{R}$ is not the only collection that is unlistable and hence larger than $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$. We can use a similar diagonalization proof to show that the **power set** of the natural numbers $\mathbb{N}$ (i.e., $\mathcal{P}(\mathbb{N})$) cannot be listed either. The cardinality of $\mathcal{P}(\mathbb{N})$ is also $\aleph_1$. There are still bigger infinities beyond this, going upwards through $\aleph_2$, $\aleph_3$, and so on.

at the start of this chapter, and it is easy enough to imagine an unlimitedly intelligent and powerful entity (let's just call it God) actually finishing that list.

But this proof tells us that if there were such a being, even God could not actually do this. The real numbers are so dense that, somehow, they are just not the sort of collection that can be listed.

37.4 Summary

IN THIS CHAPTER, we looked at the real numbers $\mathbb{R}$, and why they are not listable. By using the diagonalization trick, we were able to show that if we assume that the real numbers are listable, we can use diagonalization to construct a number that is not on the list. Hence, $\mathbb{R}$ must *not* be listable, and therefore it is bigger than $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$.

[38]

FURTHER READING

To pursue questions about infinities further, the following list may offer some helpful starting points.

- Boolos, Burgess, and Jeffrey (2002, chs. 1–2) offers a slow-paced introduction to the ideas of listability (enumerability) and the diagonalization trick.
- Stewart (1995, ch. 9) provides a good but introductory-level discussion of some of the basic concepts that underly our modern discussions of infinity.
- Stewart and Tall (2015, ch. 14) offer a slightly more advanced introduction to the mathematics of infinities.
- Steinhart (2018, ch. 9) provides a non-technical discussion not only to infinities of size $\aleph_0$, but also to the infinities that are bigger than $\aleph_0$.

- Wilder (2012, ch. 4) provides another non-technical discussion of questions around infinities.
- Zach (2019, ch. 4) provides a helpful, somewhat technical discussion of the topics of infinity, cardinality, and diagonalization.
- Cummings (2018, ch. 2) offers a slow-paced discussion of cardinality and infinities, along with how to write some of the relevant proofs.
- (Pinter, 2014, ch. 7) provides a simpler textbook-level discussion of infinities and listability, along with proofs of most of the basic facts.
- (Enderton, 1977, ch. 6) offers a textbook-level discussion of infinities, especially in the first couple sections of the chapter. Enderton also offers a different way to traverse the rational numbers so as to list them.

Part [8]

ALGEBRAS

What exactly is **algebra**? And how many are there? (There are many!) Algebra is the study of combining things inside of a set. For instance, we can take numbers and add them together to get other numbers, we can take moves in a game and sequence them to get other moves, and so on. It turns out that there are deep structural patterns in all of these, and this kind of structure is what algebra studies.

In Part 8, we look at **algebraic structures**, which are sets equipped with combining **operations** that tell us how to put things together in the set. We discuss how to "**calculate**" answers with algebraic structures, and we look at a variety of different **examples** of algebras.

[39]

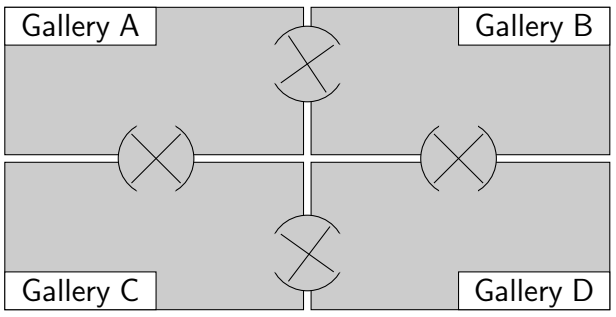
A FIRST EXAMPLE

ALGEBRA IS THE STUDY OF COMBINING THINGS inside of a set. Before we get too far into the weeds, let’s first work through an example, so that we can get a clearer picture of the kind of thing we are talking about.

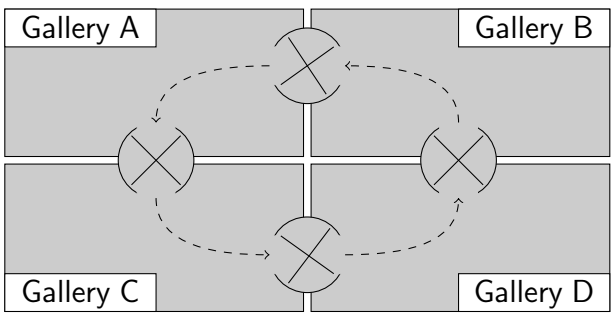
Ponder. What do you think “combining things inside a set” means? Can you think of any examples? How about combining groups of marbles from a bag? What about adding natural numbers together?

39.1 A Museum

SUPPOSE WE ARE VISITING A MUSEUM. It has four separate galleries, with revolving doors between each gallery. The galleries are laid out on a floor plan like this:

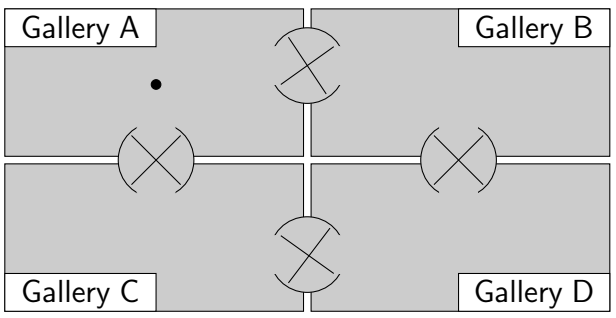


The doors only rotate one-way, so that patrons always have to walk in the same direction through the galleries. Like this:



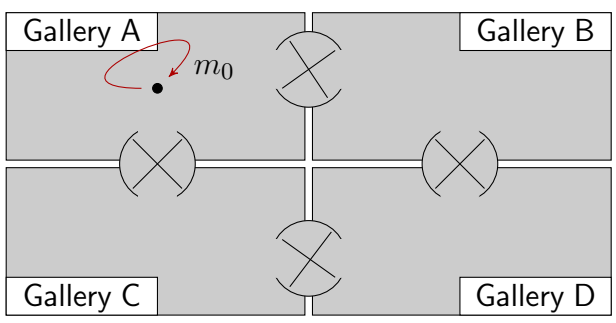
Remark 39.1. The fact that we start in gallery A is arbitrary. We could put the dot in any of the four galleries. Think of the dot not so much as "gallery A," but rather as "our current gallery."

Suppose that we are standing in gallery A, at the point marked by a dot:



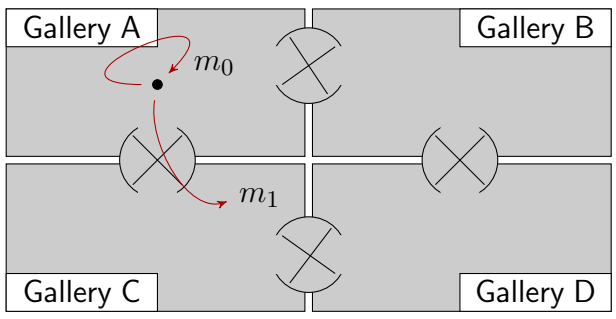
What kinds of movements can we make from this point, to get to any of the galleries? Perhaps the simplest movement

we can make is none at all: just stay in our current gallery! Let's call this movement m_0, to indicate that we move forward *zero* galleries:



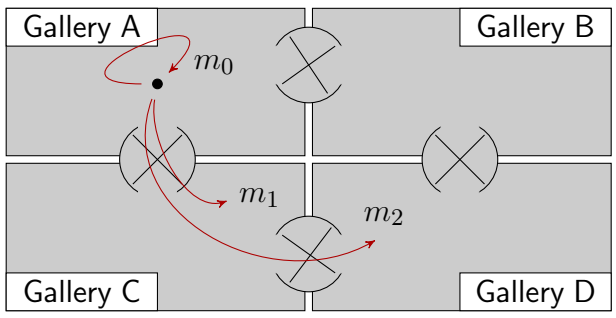
Remark 39.2. In whatever gallery we are in (be it gallery A, or B, or whatever), we can make an m_0 movement: we can just stay put in that gallery.

Another thing we can do is move to the next gallery. Let's call this m_1, to indicate that we move forward *one* gallery:



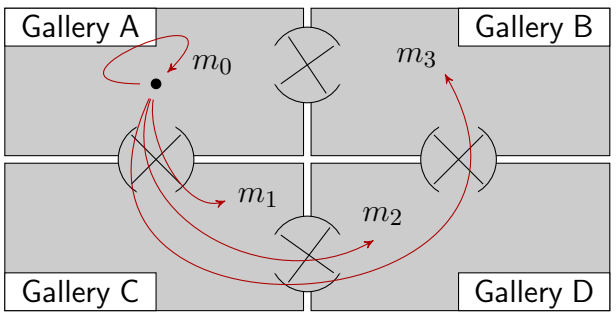
Remark 39.3. From gallery A, m_1 takes us to gallery C. But we can do the same move from any other gallery. If we were standing in gallery D, m_1 would take us to gallery B. An m_1 move is simply a move that goes from our **current** gallery, to the **next** gallery.

We can also move forward *two* galleries (m_2):



Remark 39.4. As with m_0 and m_1, m_2 and m_3 moves can be made from any of the four galleries. If we are standing in gallery D, then m_2 would take us to gallery A, and m_3 would take us to gallery C.

And of course, we can move forward *three* galleries (m_3):



At this point, we've catalogued four different moves we can make. Let's put them into a set, and let's call it A:

$$A = \{m_0, m_1, m_2, m_3\}$$

Remark 39.5. We don't need an m_5 for moving forward five galleries, because that would put us in the same place that m_1 would. Likewise, m_6 would put us in the same place as m_2, and so on.

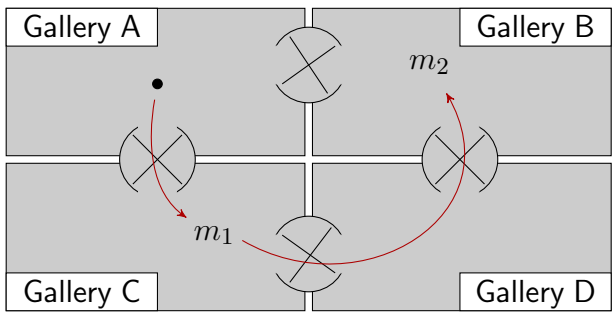
Are these all the movements we need? Should we add m_4, for moving forward *four* galleries? Well, we actually don't need m_4, for if we move forward four galleries, we end up back in the same gallery that we started in, and of course, that's exactly where m_0 (just staying put) puts us too. So we don't actually need any more movements beyond the four we already catalogued.

39.2 Combining Movements

NOW THAT WE HAVE A SET OF MOVEMENTS at hand, let's start thinking about combining them. One way to **combine** them is to just do one after the other.

Terminology. In this museum scenario, **combining** movements means performing the two movements in sequence, one after the other.

For example, from gallery A, we can do m_1, and then right after that, we can do m_2. That takes us to gallery B:



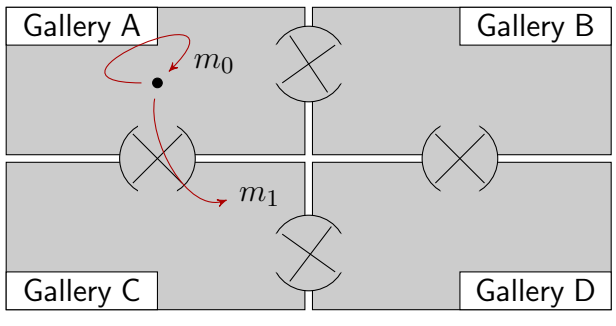
To indicate that we combine m_1 and m_2 in this way, let's write this:

$$m_1 \star m_2$$

Here's another combination: m_0 combined with m_1.

$$m_0 \star m_1$$

This means that first we do m_0 (i.e., we do nothing), then we do m_1 (i.e., we move to the next gallery). Like this:



Notation 39.1. To indicate that we **combine** the movements m_i and m_j (where "m_i" and "m_j" are any of our four possible moves), we write this: $m_i \star m_j$. Read that aloud like this: "m_i combined with m_j," or even "m_i plus m_j" or "m_i times m_j" so long as we remember that we don't mean the regular "plus" and "times" here. What we really mean is doing the one movement first, then the other movement immediately after.

39.3 Movement Arithmetic

COMBINED MOVEMENTS are equal to other movements. Think about the combined movement "$m_1 \star m_2$." If we perform m_1

then m_2, we advance a total of three galleries. Is there any other move in our set of possible moves that would take us to the same destination? Yes, there is: m_3 would also advance us three galleries, and so m_3 would take us to the same location. Hence, we can write this:

Remark 39.6. Combined movements are "equal" to other movements in our set of possible movements.

$$m_1 \star m_2 \text{ gets us to the same gallery as } m_3$$

Let's now use an equals sign (i.e., "$=$") as a shorthand for the expression "gets us to the same gallery as." Then we can rewrite the above expression like this:

$$m_1 \star m_2 = m_3$$

Notation 39.2. In this museum scenario, we are setting up a special meaning for "$=$" (the equals sign). Here, we are stipulating that "$x = y$" means "x gets us to the same gallery as y."

Read that aloud like this: "the gallery we reach by doing m_1 then m_2 is the same gallery we reach by just doing m_3."

Let's consider another case: m_0 followed by m_1.

$$m_0 \star m_1$$

Since m_0 essentially is a "do nothing" move (it moves us nowhere), following it up with an m_1 move is the same as just doing m_1 in the first place. So, we can write this:

$$m_0 \star m_1 \text{ gets us to the same gallery as } m_1$$

Or, if we rewrite it with the equals sign:

$$m_0 \star m_1 = m_1$$

You can see that we can do a kind of "arithmetic" here with these moves. We can combine two of our moves, and that will get us to the same gallery as another one of our moves.

We can write such equalities down as equations, which have the following shape:

$$\underbrace{m_i \star m_j}_{\text{the gallery we reach by doing } m_i \text{ then } m_j} = \underbrace{m_k}_{\text{the gallery we reach by doing just } m_k}$$

39.4 A Cayley Table

Let's build a kind of "multiplication table" for the "arithmetic" we are doing here. We call this kind of table a **Cayley table**. We'll start by listing the available moves down the left side, and also across the top:

Terminology. A **Cayley table** is a kind of "multiplication table," that tells us how to combine things from the same set. We list the elements from the set down the left side, and across the top, and then in each cell, we write what we get when combine the elements from that row and column.

$\star$	m_0	m_1	m_2	m_3
m_0				
m_1				
m_2				
m_3				

In each cell, we will fill in what we get when we combine the moves for that row and that column.

First, let's fill in what "$m_1 \star m_2$" is equal to. We know from above that "$m_1 \star m_2$" is equal to "m_3." So let's find "m_1" on the left side of the table and "m_2" at the top of the table, and then let's write "m_3" where that row and column meet:

$\star$	m_0	m_1	m_2	m_3
m_0				
m_1			$\mathbf{m_3}$	
m_2				
m_3				

Remark 39.7. In a Caley table, the rows and columns correspond to the two moves that we combine, and the cell where each row/column meets tells us which move the combination of that row/column is equal to. Hence, in this picture, we can see that combining m_1 and m_2 is equal to m_3, i.e., it tells us that "$m_1 \star m_2$" equals "m_3."

Whenever we want to find out what "$m_1 \star m_2$" equals, we can just look it up in this table.

Next, let's fill in what "$m_0 \star m_1$" is equal to, since we know this one too. Above, we saw that "$m_0 \star m_1$" is equal to "m_1," so let's add that in:

$\star$	m_0	m_1	m_2	m_3
m_0		$\mathbf{m_1}$		
m_1			m_3	
m_2				
m_3				

Remark 39.8. It would be a good exercise to fill out this table for yourself, on a separate piece of paper, then check it against the table here.

We can go on like this, filling in every cell. Here is the full table, with everything filled in:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

39.5 Solving Equations

NOW THAT WE HAVE A CALEY TABLE, we can **solve equations**, by looking up combinations in this table. As an example, consider the following equation:

$$m_2 \star (m_3 \star m_2) = ??$$

Notation 39.3. The parentheses in the equation tell us which combination to do first. We always solve *inside* the parentheses first.

How do we solve this? Well, we could look at the floor plans of our museum, then figure it out by drawing in the correct number of arrows. But we can also just use our Cayley table. First, we look at "$m_3 \star m_2$":

$$m_2 \quad \star \quad (\underbrace{m_3 \quad \star \quad m_2}_{\downarrow\,??})$$

To "solve" this, we look in our table. We find "m_3" on the left and "m_2" at the top, and we look where they meet:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	$\mathbf{m_1}$	m_2

So the answer to "$m_3 \star m_2$" is "m_1":

$$m_2 \quad \star \quad (\underbrace{m_3 \quad \star \quad m_2}_{\downarrow\,\mathbf{m_1}})$$

Next, we want to figure out "$m_2 \star m_1$":

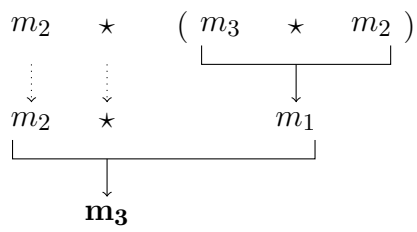
$$\begin{array}{l} m_2 \quad \star \quad (\underbrace{m_3 \quad \star \quad m_2}) \\ \underbrace{\overset{\vdots}{m_2} \quad \overset{\vdots}{\star} \qquad\qquad m_1}_{\downarrow\,??} \end{array}$$

Remark 39.9. Once we have "solved" inside the parentheses, we substitute in the answer. Then we solve what remains.

To figure out the answer to "$m_2 \star m_1$," we look in our table again. We find "m_2" on the left side and "m_1" at the top, then we look where they meet:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	$\mathbf{m_3}$	m_0	m_1
m_3	m_3	m_0	m_1	m_2

So the answer to "$m_2 \star m_1$" is "m_3":



With that, we have solved our equation:

$$m_2 \star (m_3 \star m_2) = m_3$$

Solutions. The first equation is indeed true. Here's the computation:

$$\begin{aligned} m_2 \star m_3 &= ((m_3 \star m_0) \star m_1) \star m_1 \\ m_1 &= (m_3 \star m_1) \star m_1 \\ m_1 &= m_0 \star m_1 \\ m_1 &= m_1 \end{aligned}$$

Here's a solution to the second question:

$$\begin{aligned} m_2 \star x &= m_0 \\ m_2 \star m_2 &= m_0 \\ \text{so } x &= m_2 \end{aligned}$$

Example 39.1. As a further example, see if you can figure out if this is true:

$$m_2 \star m_3 = ((m_3 \star m_0) \star m_1) \star m_1$$

Would doing the moves on the left side of the equation get us to the same gallery as doing the moves on the right side of the equation? To answer this, use the Cayley table to simplify the left side and the right side of the equation.

Here's a final example. Consider this:

$$m_2 \star x = m_0$$

Which move (m_0, m_1, m_2, or m_3) can we put in place of x to make this equation true? To solve this, use the Cayley table.

39.6 Summary

IN THIS CHAPTER, we looked at an example of "combining things inside a set." In this case, we had a set consisting of four movements that we could do to move through the galleries in a museum:

$$\{m_0, m_1, m_2, m_3\}$$

With this, we did the following:

- We noted that we can **combine** movements, in the sense that we can do one movement m_i and then immediately follow it up with another movement m_j (where m_i and m_j are any of our four possible moves). To symbolize that we combine the movements m_i and m_j in this fashion, we write: "$m_i \star m_j$."
- We also noted that when we do combine movements m_i and m_j in this fashion, the combination movement "$m_i \star m_j$" takes us to the *same location* as another one of our movements m_k (where m_k is one of our four possible moves). So, we can say that "$m_i \star m_j$" is **equal** to "m_k," which we write like this: $m_i \star m_j = m_k$.
- We can build a **Caley table**, which records every combination of two movements, and it shows us which other movement each combination is equal to.
- Using a Caley table, we can **solve equations** about which movements and combinations thereof are equal.

[40]

ALGEBRAIC STRUCTURES

IN CHAPTER 39, we looked at an example of combining movements to get around a museum. This was an example of an **algebraic structure**. In this chapter, we will define more precisely what an algebraic structure is.

Ponder. If you've done any algebra (say, in school), then think about this: in what ways did the example from Chapter 39 remind you of doing algebra?

40.1 Binary Operations

IN THE MUSEUM EXAMPLE, we have a set A of moves, which contains these four moves:

$$A = \{m_0, m_1, m_2, m_3\}$$

We can combine any two of these moves by doing one first and then the other. For instance, we can do m_1 first, then m_2.

To denote the combination of m_1 and m_2, we wrote this:

$$m_1 \star m_2$$

Every such combination of those four moves turns out to be equal to one of the other moves. We built a Cayley table, so that we can look up what every pair of moves equals:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

Let's now step back a bit and look at what we did. Notice the following:

- We took each pair of moves in our set, and we *mapped it* to it another move from our set. For example, we mapped the pair (m_1, m_2) to m_3.
- *Every pair* of moves in our set has a mapping. There are no two moves from the set that we can't look up in our Cayley table.
- Every pair of moves is mapped to *one* other move in the set, e.g., m_1 and m_2 are mapped to m_3, and to m_3 only.

Notice that when we set up our Cayley table for "$\star$," we specified a **function**. The combining operation "$\star$" is a function that maps *pairs of moves* to *individual moves*.

To see this, we can write out the function in the usual way that we write functions. The function's name/symbol is "$\star$," and here is its signature:

$$\star : A \times A \to A$$

Remark 40.1. Recall from Chapter 14 that a **function** f from one set to another maps each item in the first set to one item in the second set.

Notation 40.1. Recall from Chapter 11 that the **product** of a set A is every pair of elements from A. we write the product of A like this: $A \times A$.

Notation 40.2. We write the **signature** of a function f from a domain A to the same codomain A like this: $f : A \to A$. In this case, the domain is the product $A \times A$, so it goes from $A \times A$ to A.

Read that like this: "$\star$ is a function that maps pairs of objects from A to other objects in A." We can specify which pairs are mapped to which other objects, by writing it all out in one big function lookup table:

$\star$				
Domain	**Codomain**		**Domain**	**Codomain**
(m_0, m_0)	m_0		(m_0, m_2)	m_2
(m_1, m_0)	m_1		(m_1, m_2)	m_3
(m_2, m_0)	m_2		(m_2, m_2)	m_0
(m_3, m_0)	m_3		(m_3, m_2)	m_1
(m_0, m_1)	m_1		(m_0, m_3)	m_3
(m_1, m_1)	m_2		(m_1, m_3)	m_0
(m_2, m_1)	m_3		(m_2, m_3)	m_1
(m_3, m_1)	m_0		(m_3, m_3)	m_2

Remark 40.2. The Cayley table contains the same information as this big lookup table here, but the Cayley table is more compact. It takes up much less space, and after you get used to it, it's even easier to use. So, in algebra, we prefer to use Cayley tables instead of writing out bigger tables like this one here. Still, it is instructive to see that the Cayley table specifies the same function.

With this table, we can see that, for example, the function maps the pair (m_3, m_0) to m_3. With the usual function notation, we would write that like this:

$$\star(m_3, m_0) = m_3$$

Notation 40.3. Recall from Chapter 14 that if a function f maps x to y, we write that like this: $f(x) = y$. In this case, our function is named "$\star$" instead of f, and "x" is a pair of objects, e.g., "(m_3, m_0)."

In algebra, we usually just write the function's name (i.e., the "$\star$" symbol) in between "m_3" and "m_0," like this:

$$m_3 \star m_0 = m_3$$

Notation 40.4. We could write regular addition equations in the same two ways: either as "$+(3, 5) = 8$" or as "$3 + 5 = 8$." In math, both mean exactly the same thing, they are just different notations.

This is just a matter of notational convenience. Both ways of writing this out mean exactly the same thing.

All of this makes it clear that the combining operation "$\star$" that we defined in Chapter 39 is nothing more than a function from $A \times A$ to A.

In algebra, we call such a function a *binary operation*. So, if we have a set A, a **binary operation** is just a function that maps every pair of objects from A to some other object in A. Let's write that down as a definition:

Definition 40.1 (Binary operations)**.** Given a set A, we will say that a **binary operation** is a function $f : A \times A \to A$ that maps every pair of objects from A to another object in A. Instead of denoting the function with a letter like "f," we will denote it with a symbol such as "$\star$." Instead of writing "$\star(x, y) = z$" to indicate that $\star$ maps the pair (x, y) to z, we will write "$x \star y = z$."

Example 40.1. A binary operation needn't be as nice as the one we defined in Chapter 39. A binary operation can be **entirely made up**, if we so desire. Here's an example. Suppose we have a set B that looks like this:

$$B = \{a, b, c\}$$

Let's make up a binary operation $\star$ for B:

$\star$	a	b	c
a	b	b	b
b	b	b	b
c	b	b	b

Remark 40.3. This version of "$\star$" is a binary operation because it satisfies the definition. It maps every pair of items from B to another item in B.

This maps every pair of items from our set to b! Hence, solving equations are easy, because every combination is mapped to b. For instance:

$$a \star b = b \qquad (a \star a) \star c = b \qquad a \star (b \star c) = b$$

Also, every combination is going to turn out to be equal, because they all reduce to b. For instance:

$$a \star b = b \star a \qquad (a \star a) \star c = b \star c \qquad a \star c = a \star (b \star c)$$

Still, this version of "$\star$" is a **binary operation**, because it's a function from $B \times B$ to B.

Example 40.2. Suppose we have a set C that contains the primary colors:

$$C = \{\text{red}, \text{blue}, \text{yellow}\}$$

Let's now say that "$x \star y$" means that we mix the colors x and y to make a new color. Let's define this with the following Cayley table:

$\star$	red	blue	yellow
red	red	purple	orange
blue	purple	blue	green
yellow	orange	green	yellow

Using this table, we can say that red mixed with blue makes purple, or yellow mixed with blue makes green:

$$\text{red} \star \text{blue} = \text{purple} \qquad \text{yellow} \star \text{blue} = \text{green}$$

Is this version of "$\star$" a **binary operation**? The answer is no. It fails to be a binary operation because some of the combinations result in colors that are outside of the original set. None of purple, green, or orange are in C.

Remark 40.4. By definition, a binary operation maps pairs of items in a set to other items in the **same** set. It must point in-house, not out-of-house. In this case, "purple," "green," and "orange" are out-of-house. They are not contained in the original set C.

Example 40.3. Suppose we have a set D with the following contents:

$$D = \{\text{left}, \text{right}, \text{up}, \text{down}\}$$

Suppose we want to specify an operation "$\star$" on D with this Cayley table:

$\star$	left	right	up
left	right	down	right
right	down	left	left
down	left	right	up

Is this a **binary operation**? The answer is no, because it doesn't have a mapping for every pair of elements from D. Some mappings are missing.

Remark 40.5. By definition, a binary operation must map *every* pair of items from the set to another item in the set. This Cayley table is missing some mappings. For instance, none of these are present: $(\mathsf{up}, \mathsf{left})$, $(\mathsf{up}, \mathsf{right})$, $(\mathsf{up}, \mathsf{up})$, $(\mathsf{left}, \mathsf{down})$, $(\mathsf{right}, \mathsf{down})$, $(\mathsf{down}, \mathsf{down})$.

40.2 Algebraic Structures

WITH THE IDEA OF A BINARY OPERATION at hand, we are now ready to define an algebraic structure. An **algebraic structure** is just a base set along with one or more binary operations that we attach to it. We call the base set the **carrier set**, because it "carries" the binary operation, so to speak.

Terminology. An **algebraic structure** (or an **algebra** for short) is a base set equipped with a binary operation. The base set is called the **carrier set** of the algebra.

For example, if we have a set called "A" and a binary operation "$\star$," then we can put the two together, into a pair:

$$(A, \star)$$

Read that out loud like this: "the pair comprised of the set A and the binary operation $\star$." Or, if you like, read it like this: "the set A equipped with the binary operation $\star$."

This is an algebraic structure, because it is a set that has a binary operation we have attached to it.

We can give a name to this algebraic structure. Let's call it $\mathbf{S}$. We can write that out like this:

$$\mathbf{S} = (A, \star)$$

Remark 40.6. When we specify an algebraic structure like this, we of course need to make sure that we also specify what A and $\star$ are. Our dialogue partners always need to know which set and which binary operation we have in mind.

Read that aloud like this: "the structure $\mathbf{S}$ is defined as the pair comprised of the set A and the binary operation $\star$."

Here's another way to read it: "$\mathbf{S}$ is an algebraic structure comprised of the carrier set A equipped with the binary operation $\star$."

An algebraic structure can have more than one binary operation attached to it. For instance, imagine if we had defined another binary operation that we use the symbol "$\bullet$" for. Then we could define another algebraic structure $\mathbf{T}$ that consists of A and both operations $\star$ and $\bullet$. Like this:

$$\mathbf{T} = (A, \star, \bullet)$$

Read that aloud like this: "$\mathbf{T}$ is an algebraic structure comprised of the carrier set A equipped with the binary operations $\star$ and $\bullet$."

Let's put these ideas down in a formal definition for algebraic structures.

Definition 40.2 (Algebraic structures)**.** Given a set A and one or more binary operations $\star$, $\bullet$, ..., we will say that an **algebraic structure** (or an **algebra** for short) is the tuple $(A, \star, \bullet, \ldots)$. We will call the set A the **carrier set** of the algebra, and we will use names like $\mathbf{S}$ and $\mathbf{T}$ to denote these structures.

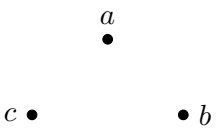
Example 40.4. Consider again the set and binary operation that we specified in Example 40.1 above. We have a set B:

$$B = \{a, b, c\}$$

And we have a binary operation $\star$:

$\star$	a	b	c
a	b	b	b
b	b	b	b
c	b	b	b

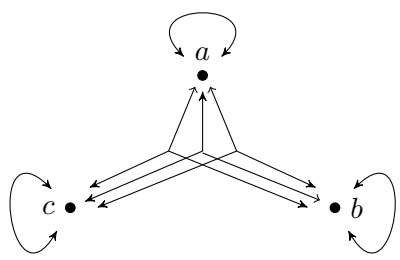
Remark 40.7. By attaching a binary operation to a set, we add **structure** to it: the operation connects pairs in the set to other objects in the set. For instance, suppose we have a set $A = \{a, b, c\}$. Without any structure, it's just free-floating dots:



But suppose we attach a binary operation $\star$, defined like this:

$\star$	a	b	c
a	a	c	b
b	c	b	a
c	b	a	c

Because we've added this on, the set now looks something like this:



That certainly has more structure than our initial free-floating dots!

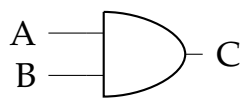
Hence, we can put together an algebraic structure from these two, which we might call "$\mathbf{S}$":

$$\mathbf{S} = (B, \star)$$

40.3 Two Operations

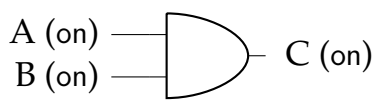
WE HAVE LOOKED AT STRUCTURES equipped with one binary operation. Let's close this chapter with an example of a structure equipped with *two* operations.

In an electronic circuit, there is a widget called an **and-gate**. It has two wires coming into it, and one wire coming out of it. Like this:

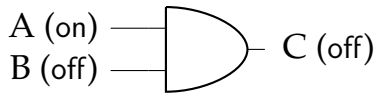


Here the input wires (the two wires going into the gate) are labeled A and B, and the output wire (the wire coming out of the gate) is labeled C.

An and-gate is designed to function in a particular way: if both of its input wires are "on" (i.e., if they each have current flowing through them), then the gate will allow the current to pass through, in which case its output wire will have current coming out of it too. Like this:

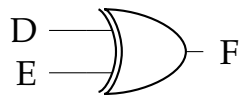


However, if one of the input wires is "off" (i.e., if it doesn't have current flowing through it), then the and-gate will not allow the current to pass through, in which case its output wire will be "off." Like this:



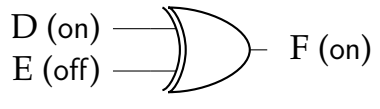
So an and-gate only lets current through if *both* of its input wires are "on."

Another similar widget is called a **xor-gate** (pronounced "zore gate"). It also has two input wires and one output wire, but we draw it with a different symbol:

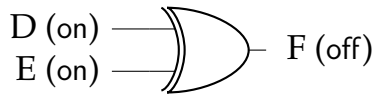


Here the input wires (the two wires going into the gate) are labeled D and E, and the output wire (the wire coming out of the gate) is labeled F.

A xor-gate is designed to allow current through only when *one* of its input wires has current flowing through it. Like this:



If both of the input wires are "on" (i.e., they have current flowing through them), then the xor-gate will not allow the current to pass through, in which case its output wire will be "off." Like this:



So a xor-gate only lets current through when *exactly one* of its input wires is "on."

Let's model all of this with algebra. Let's call our carrier set $\mathbb{B}$, and let's say it has only two elements in it:

$$\mathbb{B} = \{\mathsf{on}, \mathsf{off}\}$$

Next, let's model an **and-gate** as a binary operation called "&," with the following Cayley table:

&	on	off
on	on	off
off	off	off

With this, we can express whether the output wire of an and-gate will be on or off, like this:

$$A \,\&\, B = C$$

For instance, if A is on and B is also on, then C will also be on:

$$\mathsf{on} \,\&\, \mathsf{on} = \mathsf{on}$$

But if either of A or B is off, then C will also be off, for instance:

$$\mathsf{on} \,\&\, \mathsf{off} = \mathsf{off}$$

Now, let's model a **xor-gate** as another binary operation, called "$\oplus$," with the following Cayley table:

$\oplus$	on	off
on	off	on
off	on	off

With this, we can express whether the output wire of a xor-gate will be on or off, like this:

$$D \,\oplus\, E = F$$

For instance, if D is on and E is also on, then F will be off:

Remark 40.8. To compute the answer to "on & on," we look in our Cayley table:

&	on	off
on	**on**	off
off	off	off

So, the answer is "on." Hence, "on & on = on." To compute the answer to "on & off," we can again look in our Cayley table:

&	on	off
on	on	**off**
off	off	off

For this, the answer is "off." Hence, "on & off = off."

Remark 40.9. To compute the answer to "on $\oplus$ on," we look in our Cayley table:

$\oplus$	on	off
on	**off**	on
off	on	off

So, the answer is "off." Hence, "on $\oplus$ on = off." To compute the answer to "on $\oplus$ off," we can again look in our Cayley table:

$\oplus$	on	off
on	off	**on**
off	on	off

For this, the answer is "on." Hence, "on $\oplus$ off = on."

$$\mathsf{on} \oplus \mathsf{on} = \mathsf{off}$$

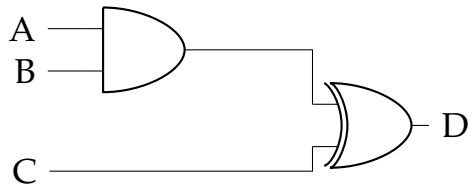
But if one of D or E is on and the other is off, then F will be on, for instance:

$$\mathsf{on} \oplus \mathsf{off} = \mathsf{on}$$

Let's pack our base set and these two binary operations together, to make an algebraic structure called B:

$$\mathbf{B} = (\mathbb{B}, \&, \oplus)$$

Now consider a more complex circuit:



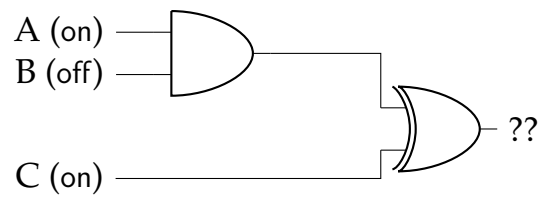
We can encode this as an equation. In the picture, A and B are combined by an **and-gate**, so we can represent that like this:

$$A \,\&\, B$$

Then the output wire of the and-gate is joined with C by a **xor-gate**, so we can represent that like this:

$$(A \,\&\, B) \oplus C$$

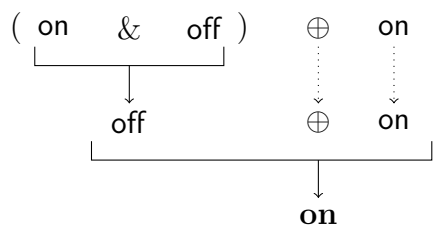
Now, let's suppose that, in our circuit, A is on, B is off, and C is on. What will the output be:



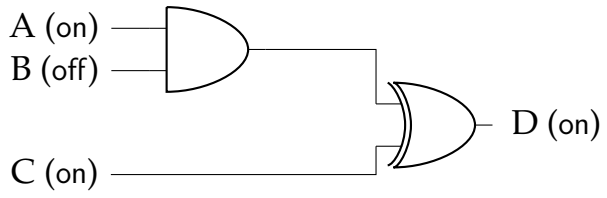
We could work the answer out by hand, by staring at the diagram and thinking it through. But, we can also just plug the inputs into our equation, and then solve that. If we substitute "on" for A, "off" for B, and "on" for C, our equation to solve is this:

$$(\text{on} \ \& \ \text{off}) \ \oplus \ \text{on} = ??$$

We can solve this simply by looking up the answers in our Cayley tables:



Hence, we can conclude that the output wire in our circuit will be on (i.e., current will pass through it):



Remark 40.10. To compute the answer to "on & off," we look in our Cayley table:

&	on	off
on	on	**off**
off	off	off

Then, to compute the answer to "off ⊕ on," we can again look in our Cayley table:

⊕	on	off
on	off	on
off	**on**	off

Hence, "(on & off) ⊕ on" comes out "on."

40.4 Summary

IN THIS CHAPTER, we looked at **algebraic structures**.

- A **binary operation** "$\star$" for a set A is a function that maps every pair of elements from A to another element in A. Thus, it has this signature: $\star : A \times A \to A$.
- An **algebraic structure** (or an "**algebra**" for short) is a tuple comprised of a base set and one or more binary operations. We call the base set the **carrier set** of the algebra.

[41]

ISOMORPHISMS

IN CHAPTER 40, we saw that an algebraic structure is a base set equipped with one or more binary operations. In this chapter, we will look at the question of **isomorphisms** for algebraic structures: when can we say that two algebraic structures have the same shape or structure?

Ponder. What do you think it would mean to say that two algebraic structures are “essentially the same,” under the hood, in their internal structure?

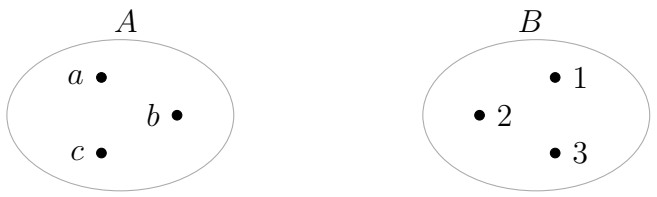
41.1 Maps between Algebras

LET’S START with how we can map one algebraic structure to another. To construct a **map** (i.e., a **function**) from one algebraic structure to another, all we need to do is map the carrier set of the one to the other.

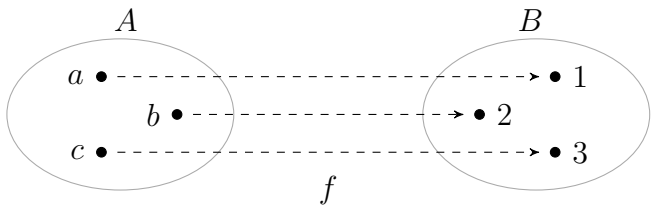
Example 41.1. Suppose we have two algebras:

$$\mathbf{S} = (A, \star) \qquad \mathbf{T} = (B, \star)$$

Let's ignore their binary operations for now. Let's just look at their carrier sets A and B. Suppose they look like this:



Let's create a map (a function) from $\mathbf{S}$ to $\mathbf{T}$, Let's call our function f. To create this, all we need to do is map each element from A to an element in B. For example, we could do this:



That is enough to map the structure $\mathbf{S}$ to $\mathbf{T}$. So, let's denote its **signature** like this:

$$f : \mathbf{S} \to \mathbf{T}$$

Read that like this: "the function f maps the algebraic structure $\mathbf{S}$ to the algebraic structure $\mathbf{T}$." Or, just read it like this: "f maps $\mathbf{S}$ to $\mathbf{T}$."

Example 41.2. Let's construct a different mapping. Let's call this one $g : \mathbf{S} \to \mathbf{T}$, and let's define it like this:

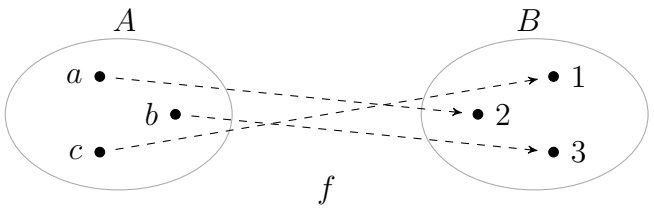
Remark 41.1. In other words, here is the mapping:

$$\begin{aligned} f(a) &= 1 \\ f(b) &= 2 \\ f(c) &= 3 \end{aligned}$$

Remark 41.2. This function maps the set A to the set B, so we could denote its signature like this:

$$f : A \to B$$

But in this context, this function is really intended as a map between our algebraic structures $\mathbf{S}$ and $\mathbf{T}$, so we write its signature as going from $\mathbf{S}$ to $\mathbf{T}$ instead of from A to B.

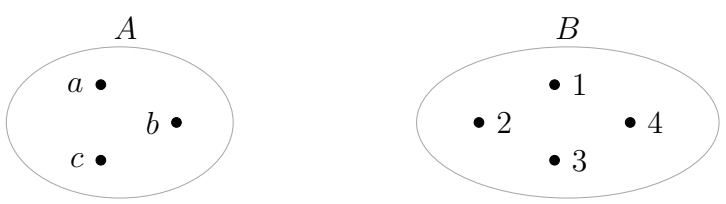


In other words, here is the mapping:

$$g(a) = 2 \qquad g(b) = 3 \qquad g(c) = 1$$

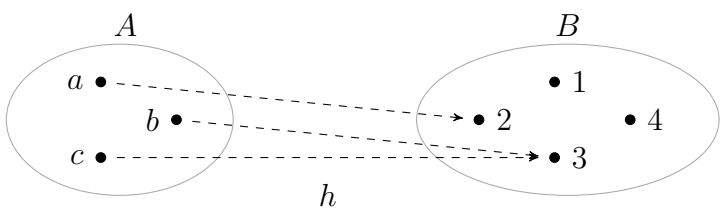
Remark 41.3. The function g differs from f because it maps elements from the first carrier set to different elements in the other carrier set.

Example 41.3. Let's look at one more example. Suppose A and B look like this:



Remark 41.4. The cardinality of A is $|A| = 3$, while the cardinality of B is $|B| = 4$.

Notice that B has more elements than A. We can still create a mapping. Let's call this map $h : \mathbf{S} \to \mathbf{T}$, and define it like this:



In other words, here is the mapping:

$$h(a) = 2 \qquad h(b) = 3 \qquad h(c) = 3$$

Remark 41.5. Recall from Chapter 16 that functions can be injective, surjective, or bijective. Here, h is neither **surjective** nor **injective**. By contrast, f and g are both injective and surjective (hence, they are **bijective**).

There are two points in B that have no arrows coming to them, namely 1 and 4. Since B is a bigger set than A, it is not possible for h to cover all of the elements in B.

41.2 Preserving Structure

NOT ALL MAPS BETWEEN ALGEBRAIC STRUCTURES are equal. Some **preserve the binary operation**, while others do not.

What does it mean to say that a map "preserves the binary operation," exactly? Suppose we have a map f from $\mathbf{S}$ to $\mathbf{T}$:

$$f : \mathbf{S} \to \mathbf{T}$$

To say that f preserves the operation means that it preserves the structure of the **Cayley table**. So, whenever $x \star y$ equals z in the first structure, then $f(x) \star f(y)$ will equal $f(z)$ in the second structure:

$$\text{if } x \star y = z \text{ in } \mathbf{S}, \text{ then } f(x) \star f(y) = f(z) \text{ in } \mathbf{T}$$

Let's write this idea down as a definition.

Definition 41.1 (Structure-preserving maps)**.** Given two algebraic structures $\mathbf{S} = (A, \star)$ and $\mathbf{T} = (B, \star)$, we will say that a map $f : \mathbf{S} \to \mathbf{T}$ is a **structure preserving** map when, for every x, y, z in A:

$$\text{if } x \star y = z \text{ in } \mathbf{S}, \text{ then } f(x) \star f(y) = f(z) \text{ in } \mathbf{T}$$

Example 41.4. Suppose we have two algebraic structures:

$$\mathbf{S} = (A, \star) \qquad \mathbf{T} = (B, \star)$$

Terminology. A map from $\mathbf{S}$ to $\mathbf{T}$ **preserves the binary operation** if whenever $x \star y = z$ in $\mathbf{S}$, then $f(x) \star f(y) = f(z)$ in $\mathbf{T}$.

Remark 41.6. The rule says:

$$\text{if } x \star y = z \text{ in } \mathbf{S}, \text{ then } f(x) \star f(y) = f(z) \text{ in } \mathbf{T}.$$

This can be written a different way. We have two equations here:

$$x \star y = z$$
$$f(x) \star f(y) = f(z)$$

Flip both around:

$$z = x \star y$$
$$f(z) = f(x) \star f(y)$$

The first tells us that another way to write "z" is "$x \star y$." So, let's take the second equation, and let's replace "z" with "$x \star y$":

$$f(x \star y) = f(x) \star f(y)$$

This is an equivalent way of writing the same rule.

where the sets A and B are defined like this:

$$A = \{a, b, c\} \qquad\qquad B = \{1, 2, 3\}$$

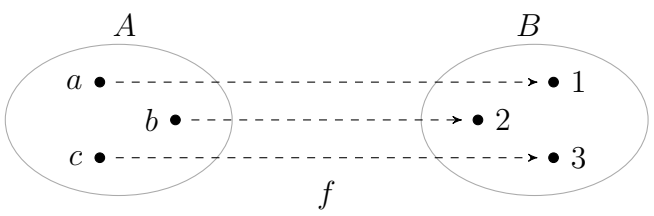
And the Cayley tables are defined like this:

$\star$	a	b	c
a	a	a	a
b	b	b	b
c	c	c	c

$\star$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

Ponder. It is worth pausing for a moment to look at these two Cayley tables. Can you see how they are similar?

Now let's define a map $f : \mathbf{S} \to \mathbf{T}$ that looks like this:



Remark 41.7. In other words, here is the mapping:

$$f(a) = 1$$
$$f(b) = 2$$
$$f(c) = 3$$

Is f a structure-preserving map? To check, we need to make sure that whenever $x \star y = z$ in $\mathbf{S}$, then $f(x) \star f(y) = f(z)$ in $\mathbf{T}$. We could manually check every possible pair of x and y. For instance, we could take a and b, and check that $a \star b$ and $f(a) \star f(b)$ correspond. If we look in $\mathbf{S}$'s Cayley table, we see that in $\mathbf{S}$:

$$a \star b = a$$

Since $f(a) = 1$, $f(b) = 2$, we would then expect this to be true in $\mathbf{T}$:

$$1 \star 2 = 1$$

And indeed, if we look in $\mathbf{T}$'s Cayley table, we can see that this is so.

It is tedious to check every pair though. There's an easier way. We can look at the two Cayley tables, and make

sure that they have the same structure (i.e., corresponding elements appear in corresponding places).

Let's do it. First, since f maps a to 1, let's make sure that a and 1 appear in the same places in their respective Cayley tables. Here they are:

$\star$	a	b	c
a	a	a	a
b	b	b	b
c	c	c	c

$\star$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

Next, let's check b and 2:

$\star$	a	b	c
a	a	a	a
b	b	b	b
c	c	c	c

$\star$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

Finally, let's check c and 3:

$\star$	a	b	c
a	a	a	a
b	b	b	b
c	c	c	c

$\star$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

We can see that in all of these cases, corresponding elements appear in corresponding positions, so f does indeed preserve the binary operation.

Example 41.5. Let's look at an example of a map that fails to preserve the binary operation. Suppose we have two algebraic structures:

$$\mathbf{S} = (A, \star) \qquad \mathbf{T} = (B, \star)$$

Remark 41.8. To say that the corresponding elements appear in the corresponding positions means both that they live in the same row and column *headers*, and they live in the same *cells*. Suppose a and 1 were positioned like this:

$\star$	b	c	a
b	a	a	a
c	b	b	b
a	c	c	c

$\star$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

Then they would *not* correspond (notice how a and 1 are in different places in the row and column headers).

where the sets A and B are defined as before:

$$A = \{a, b, c\} \qquad\qquad B = \{1, 2, 3\}$$

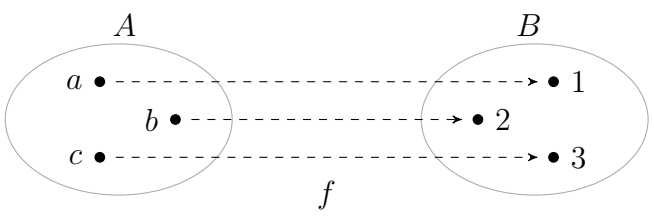
but their Cayley tables are defined like this:

$\star$	a	b	c
a	a	a	a
b	b	b	b
c	c	c	c

$\star$	1	2	3
1	1	2	3
2	1	2	3
3	1	2	3

Ponder. Just by looking at these two Cayley tables, can you see how they are different in structure?

Suppose now that we have a map $f : \mathbf{S} \to \mathbf{T}$ that is defined similar to before, like this:



Remark 41.9. In other words, here is the mapping:

$$f(a) = 1$$
$$f(b) = 2$$
$$f(c) = 3$$

Is f a structure-preserving map? To check, let's compare the Cayley tables. First, let's check a and 1:

$\star$	a	b	c
a	a	a	a
b	b	b	b
c	c	c	c

$\star$	1	2	3
1	1	2	3
2	1	2	3
3	1	2	3

We can see immediately that a and 1 occupy very different positions in their respective Cayley tables. And this is enough to show us that f does not preserve the binary operation between these two algebraic structures.

Indeed, it is easy to find an example where $x \star y = z$ in $\mathbf{S}$ but $f(x) \star f(y) \neq f(z)$ in $\mathbf{T}$. For instance, in $\mathbf{S}$, this holds:

$$a \star b = a$$

Since $f(a) = 1$ and $f(b) = 2$, we would expect this to hold in $\mathbf{T}$:

$$1 \star 2 = 1$$

But it does not hold in $\mathbf{T}$. According to $\mathbf{T}$'s Cayley table, this is actually what is correct:

$$1 \star 2 = 2$$

So for these two algebraic structures, f is not a structure preserving map.

Ponder. We have shown that f is not a structure preserving map from $\mathbf{S}$ to $\mathbf{T}$. Do you think it's possible to construct some other map from $\mathbf{S}$ to $\mathbf{T}$ that *is* structure preserving? Why or why not?

41.3 Algebraic Isomorphisms

WE CAN NOW DEFINE ISOMORPHISMS for algebraic structures. An **isomorphism** between algebraic structures is simply a *bijective*, *structure preserving* map. Let's put this down as a definition:

Definition 41.2 (Algebraic Isomorphism)**.** Given two algebraic structures $\mathbf{S} = (A, \star)$ and $\mathbf{T} = (B, \star)$, we will say that a map $f : \mathbf{S} \to \mathbf{T}$ is an **isomorphism** under two conditions: (1) f is a structure preserving map, and (2) f is bijective.

Terminology. Recall that two structures are **isomorphic** if they have the same structure "under the hood," and differ only on the surface, in the names of their elements. A function that translates one structure to another, and preserves all of the structure, is called an **isomorphism**. For sets, an isomorphism is simply any bijective function (see Chapter 17). For graphs, an isomorphism is a bijective function that preserves all the connections (see Chapter 26). For algebraic structures, it is a bijective function that preserves the structure of the binary operation.

Why do we say that f must be bijective? We add this condition because constructing a structure preserving map from $\mathbf{S}$ to $\mathbf{T}$ is not enough to guarantee that $\mathbf{S}$ and $\mathbf{T}$ have *exactly* the same structure, under the hood.

After all, if $\mathbf{T}$ is bigger than $\mathbf{S}$, we still might be able to map $\mathbf{S}$ into a smaller part of $\mathbf{T}$, and preserve the binary operation in that smaller part of $\mathbf{T}$.

But in such a case, even though f is a structure-preserving map, $\mathbf{S}$ and $\mathbf{T}$ are clearly not isomorphic. On the contrary, $\mathbf{T}$ is bigger, so there are more parts to it!

If we say that f must be structure preserving and *also* bijective, then we can guarantee that it will be an isomorphism. For a bijective function can only be constructed between two sets that are exactly the same size.

Hence, if we can construct a bijective, structure preserving map between $\mathbf{S}$ and $\mathbf{T}$, then that means their binary operations are parallel, and neither one has more elements than the other. The two structures are exact twins, apart from the names of their elements.

Whenever two algebraic structures are twins like this, we say they are **isomorphic**, and we denote that like this:

$$\mathbf{S} \cong \mathbf{T}$$

Let's put this down in a definition.

Terminology. If we can map an algebraic structure $\mathbf{S}$ into a smaller part of a bigger structure $\mathbf{T}$ and still preserve the structure of the binary operation in that smaller part of $\mathbf{T}$, then we might say that we can **embed** $\mathbf{S}$ in $\mathbf{T}$.

Definition 41.3 (Isomorphic Algebraic Structures)**.** Given two algebraic structures $\mathbf{S} = (A, \star)$ and $\mathbf{T} = (B, \star)$, we will say that $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic** when we can construct an isomorphism $f : \mathbf{S} \to \mathbf{T}$ between them. To denote that $\mathbf{S}$ and $\mathbf{T}$ are isomorphic, we will write this: $\mathbf{S} \cong \mathbf{T}$.

Terminology. If it is possible to construct an **isomorphism** between two algebraic structures $\mathbf{S}$ and $\mathbf{T}$, then we say $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic**, and we write that like this: $\mathbf{S} \cong \mathbf{T}$.

Example 41.6. The algebraic structures $\mathbf{S}$ and $\mathbf{T}$ from Example 41.4 are isomorphic. As we saw in that example, the function f is a structure preserving map because it preserves the structure of their Cayley tables. But f is also bijective: it maps

each element from A to an element in B in an exact, one-to-one, reversible way.

Thus, f is an isomorphism, and because we were able to construct an isomorphism between $\mathbf{S}$ and $\mathbf{T}$, we can conclude that they are isomorphic: $\mathbf{S} \cong \mathbf{T}$. These two structures are essentially the very same, apart from the names of their elements.

41.4 Summary

IN THIS CHAPTER, we looked at **isomorphisms** between algebraic structures.

- If we have two algebraic structures $\mathbf{S}$ and $\mathbf{T}$, we can construct a **map** (i.e., a **function**) f from $\mathbf{S}$ to $\mathbf{T}$ simply by mapping the carrier set of $\mathbf{S}$ to the carrier set of $\mathbf{T}$. The signature of f is this: $f : \mathbf{S} \to \mathbf{T}$.
- A map $f : \mathbf{S} \to \mathbf{T}$ is a **structure preserving** map if it preserves the binary operation: if $x \star y = z$ in $\mathbf{S}$, then $f(x) \star f(y) f(z)$ in $\mathbf{T}$.
- We can check if a map $f : \mathbf{S} \to \mathbf{T}$ is structure preserving by checking the Cayley tables of the two structures. If f does preserve the structure, then corresponding elements will appear in corresponding positions in the two tables.
- An **isomorphism** is a bijective, structure preserving map between structures. If we can construct an isomorphism between structures $\mathbf{S}$ and $\mathbf{T}$, then we may conclude that $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic**, i.e., they are essentially the same, apart from the names of their elements. To denote that $\mathbf{S}$ and $\mathbf{T}$ are isomorphic, we write this: $\mathbf{S} \cong \mathbf{T}$.

[42]

PROPERTIES OF OPERATIONS

IN THIS CHAPTER, we'll look at some of the properties that **binary operations** can have. In the next chapters, we'll use these properties to distinguish different kinds of algebras.

42.1 Associativity

A BINARY OPERATION $\star$ on a set A is **associative** if this is true for all elements in A:

$$(x \star y) \star z = x \star (y \star z)$$

Terminology. An operation is **associative** if "$(x \star y) \star z$" always equals "$x \star (y \star z)$." In other words, where we put the parentheses doesn't matter.

The basic idea here is just this. If we are combining x, y, and z, it doesn't matter if we first combine x and y, or if we first combine y and z. Either way comes to the same answer.

If an algebraic structure has a binary operation that is associative, then we can say that the *algebraic structure* is associative. That is, we say it is an **associative algebra**.

Definition 42.1 (Associative algebra)**.** For any algebraic structure $\mathbf{S} = (A, \star)$, we will say that $\mathbf{S}$ is an **associative algebra** if, for every x, y, and z in A:

$$(x \star y) \star z = x \star (y \star z)$$

Example 42.1. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ where the set A is defined like this:

$$A = \{a, b\}$$

and the Cayley table is defined like this:

$\star$	a	b
a	a	b
b	b	a

Is this an associative algebra? Let's check some cases. First, let's take a, b, and a, and let's check if this holds:

$$(a \star b) \star b = a \star (b \star b)$$

That is to say, does $(a \star b) \star b$ gives us the same answer as $a \star (b \star b)$? Using our Cayley table, we can reduce both sides of the equation, and we see that they do in fact come to the same thing:

Remark 42.1. You may recognize associativity from addition. Regular **addition** is associative. It doesn't matter where we put the parentheses. For example, consider these two equations:

$$2 + (3 + 4) \quad \text{and} \quad (2 + 3) + 4)$$

These give the same answer:

$$\begin{array}{cccc|ccc} 2 & + & (3+4) & & (2+3) & + & 4 \\ & & \downarrow & & \downarrow & & \\ 2 & + & 7 & & 5 & + & 4 \\ & \downarrow & & & & \downarrow & \\ & 9 & & & & 9 & \end{array}$$

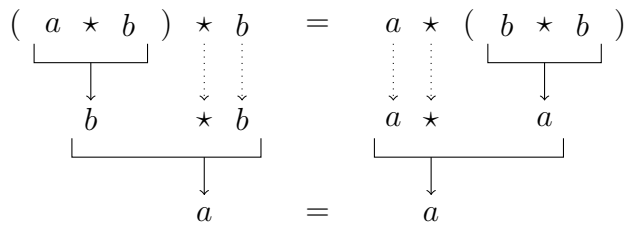
So, when we are adding number like this, where we put the parentheses doesn't matter. We still get the same answer. By contrast, **subtraction** is not associative. 'Consider these two equations:

$$2 - (3 - 4) \quad \text{and} \quad (2 - 3) - 4$$

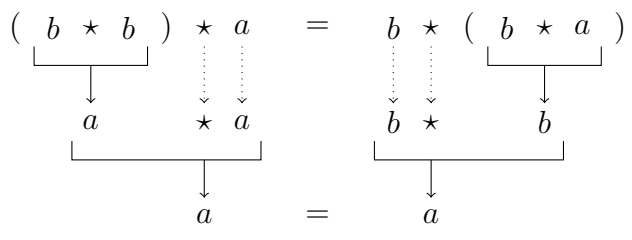
These do not give the same answer:

$$\begin{array}{cccc|ccc} 2 & - & (3-4) & & (2-3) & - & 4 \\ & & \downarrow & & \downarrow & & \\ 2 & - & (-1) & & (-1) & - & 4 \\ & \downarrow & & & & \downarrow & \\ & 3 & & & & -5 & \end{array}$$

So, when we're subtracting numbers like this, where we put the parentheses *does* matter. Subtraction is *not* associative.



Let's check another case. Let's check if $(b \star b) \star a$ gives us the same answer as $b \star (b \star a)$. If we reduce both sides of the equation, we can see that we get the same result:



If you check all possible combinations of three elements from A, you will see that we always get the same answer, no matter where we put the parentheses. So this is an associative algebra.

Remark 42.2. To show that an algebraic structure is **associative**, we have to show that parentheses do not matter in *every* possible combination of letters. If there is even one combination where the location of the parentheses matters, then the structure is *not* associative.

Example 42.2. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ where the set A is defined like this:

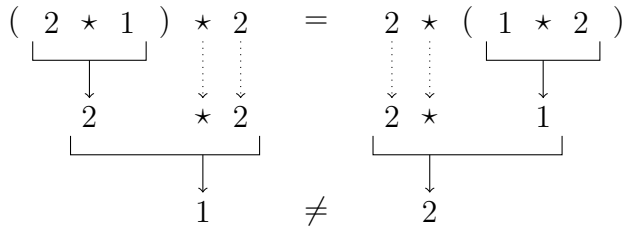
$$A = \{1, 2\}$$

and the Cayley table is defined like this:

$\star$	1	2
1	1	1
2	2	1

Remark 42.3. To show that an algebraic structure is **not associative**, all we need to do is find one case where the location of the parentheses matters.

Is this an associative algebra? It is not. Here's a case where we get a different answer if we put the parentheses in a different place:



So this is not an associative algebra, because the binary operation is not associative.

42.2 Commutativity

Terminology. An algebraic structure is **commutative** (or synonymously, **abelian**) if "$x \star y$" and "$y \star x$" always give the same answer. In other words, the order of x and y does not matter: x can come first, or y can come first, but either way, we get the same answer.

A BINARY OPERATION $\star$ on a set A is **commutative** (or synonymously, **abelian**) if this is true for all elements in A:

$$x \star y = y \star x$$

In other words, the order of x and y doesn't matter. If an algebraic structure has a binary operation that is commutative, then we can say it as a **commutative** (**abelian**) **algebraic structure**, or a **commutative** (**abelian**) **algebra**.

Definition 42.2 (Commutative algebra)**.** For any algebraic structure $\mathbf{S} = (A, \star)$, we will say that $\mathbf{S}$ is a **commutative algebra** (or synonymously, an **abelian algebra**) if, for every x and y in A:

$$x \star y = y \star x$$

Example 42.3. Consider the algebraic structure we built in Chapter 39. There we defined an algebraic structure $\mathbf{S} = (A, \star)$ where the set A was defined like this:

$$A = \{m_0, m_1, m_2, m_3\}$$

and the Cayley table was defined like this:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

Is this algebraic structure commutative? Let's check some cases. Here's one:

$$\begin{array}{ccc} \underbrace{m_1 \star m_3} & = & \underbrace{m_3 \star m_1} \\ \downarrow & & \downarrow \\ m_0 & = & m_0 \end{array}$$

We can see from this that $m_1 \star m_3$ comes out to the same as $m_3 \star m_1$, so in this case, the order of m_1 and m_3 doesn't matter.

We could check more cases, but there's an easier way to tell if an algebra is commutative. You can look at the Cayley table, and check every diagonal going from the bottom left to the top right. Each such diagonal row should have the same elements in it.

In our Cayley table, we can see that this is so. For instance, look at this diagonal:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

Remark 42.4. You may be familiar with commutativity through addition as well. Regular **addition** is commutative (abelian). Consider these two equations:

$$3 + 4 \quad \text{and} \quad 4 + 3$$

These give the same answers:

$$\begin{array}{c|c} 3+4 & 4+3 \\ \downarrow & \downarrow \\ 7 & 7 \end{array}$$

It doesn't matter if we put the "3" first or the "4" first. Either way, we get the same answer. By contrast, **subtraction** is not commutative. Consider these two equations:

$$3 - 4 \quad \text{and} \quad 4 + 3$$

These give different answers:

$$\begin{array}{c|c} 3-4 & 4-3 \\ \downarrow & \downarrow \\ -1 & 1 \end{array}$$

Remark 42.5. We can see that every highlighted cell contains m_3. The same goes for every diagonal going from the bottom left to the top right: each diagonal is filled with the same value. So this algebra is **commutative** (it is an **abelian algebra**).

Example 42.4. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ where the set A is defined like this:

$$A = \{1, 2\}$$

and the Cayley table is defined like this:

$\star$	1	2
1	1	1
2	2	1

Is this algebraic structure commutative? Let's check some cases. Here's one:

$$\underbrace{1 \star 2}_{\downarrow\ 1} \quad = \quad \underbrace{2 \star 1}_{\downarrow\ 2}$$

$$1 \quad \neq \quad 2$$

We can see here that the order does matter. We get a different answer if we put the 2 first, or the 1 first. So this is *not* a commutative (abelian) algebra.

42.3 Identity Elements

A BINARY OPERATION $\star$ on a set A has an **identity** element (or synonymously, a **unit** element) if there is some element (let's just call it e) in A that has no effect when we combine it with other elements. If we combine it with x (i.e., if we calculate "$e \star x$" or "$x \star e$"), it has no effect, and we just get back x:

$$e \star x = x \qquad \text{and} \qquad x \star e = x$$

Terminology. An algebraic structure has an **identity** element (or synonymously, a **unit** element), if there is an element e that has no effect on the combination: "$x \star e$" and "$e \star x$" always equal "x." When we are writing in pure symbols, mathematicians often use the letter "e" as a symbol for the identity element.

If an algebraic structure has a binary operation with an identity element (unit element), then we can say that the *algebraic structure* has an **identity** (or synonymously, a **unit**).

Definition 42.3 (Algebra with an identity (unit))**.** For any algebraic structure $\mathbf{S} = (A, \star)$, we will say that $\mathbf{S}$ has an **identity** (or synonymously, a **unit**) if, there is an element e in A which is such that for every x in A:

$$e \star x = x \qquad \text{and} \qquad x \star e = x$$

Example 42.5. Consider again the algebraic structure from the museum $\mathbf{S} = (A, \star)$, where $A = \{m_0, m_1, m_2, m_3\}$ and its Cayley table is this:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

Does this algebraic structure have an identity (unit) element? Yes, it's m_0. After all, no matter what we combine m_0 with, it has no effect on the combination:

$$m_0 \star m_1 = m_1 \qquad m_0 \star m_2 = m_2 \qquad m_0 \star m_3 = m_3$$
$$m_1 \star m_0 = m_1 \qquad m_2 \star m_0 = m_2 \qquad m_3 \star m_0 = m_3$$
$$m_0 \star m_0 = m_0$$

There is an easy way to check if an algebra has an identity unit. You look at the Cayley table, and you see if there is an element whose row and column copies the row and column headers exactly.

Remark 42.7. When it comes to regular **addition**, "0" (the number zero) is the identity (or unit). If you add "0" to "7," the "0" doesn't change anything. You still get "7." If you add "0" to "100," again the "0" doesn't change anything. You still get "100." Adding "0" to a number has no effect on the result. So "0" is the identity (or unit) for addition.

Similarly, when it comes to **multiplication**, "1" (the number one) is the identity (or unit). If you multiply "7" by "1," the "1" doesn't change anything. You still get "7." If you multiply "100" by "1," you still get "100." The "1" didn't change anything. So multiplying by "1" has no effect on the result. Hence, "1" is the identity (or unit) for multiplication.

In this Cayley table, let's put the row and column headers in bold, so we can see them clearly:

$\star$	$\mathbf{m_0}$	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$
$\mathbf{m_0}$	m_0	m_1	m_2	m_3
$\mathbf{m_1}$	m_1	m_2	m_3	m_0
$\mathbf{m_2}$	m_2	m_3	m_0	m_1
$\mathbf{m_3}$	m_3	m_0	m_1	m_2

Is there an element whose row and column copies these headers exactly? Yes, it's m_0. If we look at the row and column for m_0, we can see that they copy the headers exactly:

$\star$	$\mathbf{m_0}$	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$
$\mathbf{m_0}$	m_0	m_1	m_2	m_3
$\mathbf{m_1}$	m_1	m_2	m_3	m_0
$\mathbf{m_2}$	m_2	m_3	m_0	m_1
$\mathbf{m_3}$	m_3	m_0	m_1	m_2

Remark 42.8. Notice how m_0's row matches the column headers exactly:

$\star$	$\mathbf{m_0}$	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$
$\mathbf{m_0}$	m_0	m_1	m_2	m_3
$\mathbf{m_1}$	m_1	m_2	m_3	m_0
$\mathbf{m_2}$	m_2	m_3	m_0	m_1
$\mathbf{m_3}$	m_3	m_0	m_1	m_2

And m_0's column matches the row headers exactly:

$\star$	$\mathbf{m_0}$	$\mathbf{m_1}$	$\mathbf{m_2}$	$\mathbf{m_3}$
$\mathbf{m_0}$	m_0	m_1	m_2	m_3
$\mathbf{m_1}$	m_1	m_2	m_3	m_0
$\mathbf{m_2}$	m_2	m_3	m_0	m_1
$\mathbf{m_3}$	m_3	m_0	m_1	m_2

Example 42.6. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ with $A = \{1, 2, 3\}$ and a Cayley table like this:

$\star$	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

Does this algebraic structure have an identity element? To check, we look at the row and column headers. Here they are, in bold:

$\star$	**1**	**2**	**3**
1	2	3	1
2	3	1	2
3	1	2	3

Next, we look to see if there is an element whose row and column match the headers exactly. Is there one? Yes, it's 3. Here is the row and column for 3, highlighted:

$\star$	**1**	**2**	**3**
1	2	3	1
2	3	1	2
3	1	2	3

42.4 Inverse Elements

A BINARY OPERATION $\star$ on a set A has **inverses** if every element has an "opposite" element. Two elements are opposites if they cancel each other out. That is, if you combine them, you just get back the identity (unit) element.

Terminology. An algebraic structure has **inverses** if every element has an opposite. Elements are opposites if they cancel each other out, i.e., if combining them gives you back the identity element.

For convenience, let's denote the "opposite" element of x like this:

$$x^{-1}$$

Read that out loud as "the inverse of x." With this notation at hand, we can assert that an element x combined with its inverse x^{-1} gives back the identity e by writing this:

$$x \star x^{-1} = e \qquad \text{and} \qquad x^{-1} \star x = e$$

Ponder. Can an algebraic structure have **inverses** if it doesn't have an **identity** (unit) element?

Read that out loud like this: "x combined with its inverse x^{-1} equals the identity e, and also the inverse x^{-1} combined with x equals the identity e."

If every element in an algebraic structure has an inverse, then we say it is an algebraic structure with **inverses**.

Definition 42.4 (Algebra with inverses)**.** For any algebraic structure $\mathbf{S} = (A, \star)$, we will say that $\mathbf{S}$ has **inverses** if, for every x in A:

$$x \star x^{-1} = e \qquad \text{and} \qquad x \star x^{-1} = e$$

Remark 42.10. You may be familiar with inverses from addition. Regular **addition** has inverses. Take any number, e.g., "7," or "100." What is its opposite? What "cancels it out"? It's the negative version of itself. The opposite of "7" is "-7," because if we add "7" and "-7," we get "0," which is the identity (unit) for addition. Similarly, what's the inverse of "100"? It's "-100," because if we add "100" and "-100," we get "0," the identity (unit). In regular addition, for any number x, its inverse is $-x$, because "$x + (-x)$" always equals "0." So addition has inverses.

Remark 42.11. With regular addition, it's easy to figure out the inverse for any number (just reverse its positive/negative sign). But not all structures will have inverses that are negative. In this structure $\mathbf{S} = (A, \star)$, there simply are no negative numbers! The inverse of 1 is 2, the inverse of 2 is 1, and the inverse of 3 is 3.

Example 42.7. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ with $A = \{1, 2, 3\}$ and a Cayley table like this:

$\star$	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

Does this algebraic structure have inverses? To check, we need to find an opposite for each element in A. The set A contains 1, 2, and 3, so let's see if we can find an opposite for each one.

First, let's look at 1. Recall that the identity for this algebraic structure is 3. So, is there an element that we can combine with 1 to get 3? Yes, it is 2:

$$1 \star 2 = 3 \qquad \text{and} \qquad 2 \star 1 = 3$$

We can see it in the Cayley table:

$\star$	1	**2**	3
1	2	**3**	1
2	3	1	2
3	1	2	3

$\star$	**1**	2	3
1	2	3	1
2	**3**	1	2
3	1	2	3

What about 2? Does 2 have an inverse? Yes, it's 1:

$$2 \star 1 = 3 \qquad \text{and} \qquad 1 \star 2 = 3$$

How about 3? Does 3 have an inverse? Well, this one's easy, because 3 is the identity element, and the identity element combined with anything gives us back the identity element. So its inverse is just itself. It's 3!

$$3 \star 3 = 3 \qquad \text{and} \qquad 3 \star 3 = 3$$

Since every element in this algebraic structure has an opposite, we can conclude that this is an algebra with inverses.

Remark 42.12. The **identity** (unit) element is always **its own inverse**. Why? Because $e \star e$ is always e.

42.5 Summary

IN THIS CHAPTER, we looked at some **properties** of binary operations.

- If we have an algebraic structure $\mathbf{S} = (A, \star)$ and for any three elements x, y, and z in A, if $(x \star y) \star z$ gives us the same answer as $x \star (y \star z)$, then $\mathbf{S}$ is an **associative** algebra.
- If we have an algebraic structure $\mathbf{S} = (A, \star)$ and for any elements x and y in A, if $x \star y$ gives us the same answer as $y \star x$, then $\mathbf{S}$ is a **commutative** algebra (or synonymously, an **abelian** algebra).
- If an element in an algebra has no effect on combinations, i.e. if $x \star e = x$ and $e \star x = x$, then we say that e is the **identity** element (or synonymously, the **unit** element) of the algebra.
- If we have an algebraic structure $\mathbf{S} = (A, \star)$ and $\mathbf{S}$ has an identity (unit) element, then we say that $\mathbf{S}$ is an algebra **with an identity** (or **unit**).

- An element x has an opposite element (which we denote as x^{-1}) if combining the two cancels each other out, and we get back the identity (unit) element. We call the opposite of x the **inverse** of x.
- If we have an algebraic structure $\mathbf{S} = (A, \star)$ and every x in A has an inverse x^{-1}, then we say that $\mathbf{S}$ is an algebra **with inverses**.

[43]

GROUPOIDS, SEMIGROUPS, MONOIDS

In Chapter 42, we looked at some of the properties that binary operations can have. We can use those properties to sort algebraic structures into different types. In this chapter, we will look at three types of algebras: **groupoids**, **semigroups**, and **monoids**.

43.1 Groupoids

The simplest kind of algebraic structure is just a base set equipped with a single binary operation. This is called a **groupoid**.

Definition 43.1 (Groupoid)**.** For any algebraic structure $\mathbf{S} = (A, \star)$ comprised of a carrier set A and a single binary operation $\star$, we will say that $\mathbf{S}$ is a **groupoid**.

Terminology. A **groupoid** is just a carrier set equipped with a single binary operation.

Most of the examples we have looked at so far have been equipped with only a single binary relation, and so all of those qualify as groupoids.

Example 43.1. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ where the set A is defined like this:

$$A = \{1, 2\}$$

and the Cayley table is defined like this:

$\star$	1	2
1	1	1
2	2	1

Remark 43.1. A good exercise would be to check whether this groupoid is associative, or commutative. (It is neither, but can you find examples of combinations that prove it?)

This is a **groupoid** because it is comprised of a carrier set equipped with a single binary operation.

Example 43.2. Consider the integers $\mathbb{Z}$ and the binary operation of subtraction (i.e., "$-$"). Together, these make up an algebraic structure:

$$\mathbf{Z} = (\mathbb{Z}, -)$$

The carrier set is the set of integers:

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

The Cayley table is infinitely large (since there are infinitely many integers, and we can subtract any two of them). Here is part of it:

$-$	$\dots$	-1	0	1	2	3	$\dots$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$
-1	$\dots$	0	-1	-2	-3	-4	$\dots$
0	$\dots$	1	0	-1	-2	-3	$\dots$
1	$\dots$	2	1	0	-1	-2	$\dots$
2	$\dots$	3	2	1	0	-1	$\dots$
3	$\dots$	4	3	2	1	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Nobody could actually write down such a large table in one place. But let's pretend for a moment that somebody did. We could then forget everything we learned about subtraction, and just look up answers in the Cayley table. Suppose we didn't know how to compute this:

$$2 - (-1) = ??$$

We could just look it up in the table:

$-$	$\dots$	-1	0	1	2	3	$\dots$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$
-1	$\dots$	0	-1	-2	-3	-4	$\dots$
0	$\dots$	1	0	-1	-2	-3	$\dots$
1	$\dots$	2	1	1	-1	-2	$\dots$
2	$\dots$	$\mathbf{3}$	2	1	0	-1	$\dots$
3	$\dots$	4	3	2	1	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Remark 43.2. This Cayley table is too big to memorize, or even write down. One way to cope with this is to come up with some procedure that lets us *calculate* what the answer will be at any point in the table. Then we don't need to write the table down. We can just use our procedure to figure out what the relevant cell would tell us if we *could* look it up in the table.

In school, most children learn some procedure to calculate the answer to simple integer subtraction problems. For instance, they learn how to take away blocks from a bigger pile of blocks. We can look at this as just a way to tell us what the answer is in the Cayley table.

This structure $\mathbf{Z} = (\mathbb{Z}, -)$ qualifies as a groupoid, because it is comprised of a carrier set and a single binary operation.

43.2 Semigroups

BEYOND GROUPOIDS, the next simplest type of algebra is called a semigroup. A **semigroup** is a groupoid, whose binary operation is associative. So, a semigroup is a structure comprised of a carrier set equipped with one binary operation, and that binary operation is associative.

Definition 43.2 (Semigroup)**.** For any algebraic structure $\mathbf{S} = (A, \star)$ comprised of a carrier set A and a single binary operation $\star$, if $\star$ is assocative, then we will say that $\mathbf{S}$ is a **semigroup**.

Terminology. A structure is a **semigroup** if it is comprised of a carrier set and a single binary operation that is associative.

A semigroup is also a groupoid, since it satisfies the definition: it is a carrier set equipped with a single binary operation. Since its binary operation is associative, we can also call it an **associative groupoid**.

Example 43.3. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ with $A = \{1, 2, 3\}$ and a Cayley table like this:

$\star$	1	2	3
1	2	3	1
2	3	1	2
3	1	2	3

This is a semigroup, because it satisfies the definition. It is comprised of a base set equipped with a single binary operation, and that binary operation is associative. For instance, you can check that these hold:

$$(1 \star 2) \star 3 = 1 \star (2 \star 3) \qquad (2 \star 2) \star 1 = 2 \star (2 \star 1)$$
$$(1 \star 1) \star 3 = 1 \star (1 \star 3)$$

Example 43.4. Consider the integers $\mathbb{Z}$ and the binary operation of addition (i.e., "+"). Together, these make up an algebraic structure:

$$\mathbf{Z} = (\mathbb{Z}, +)$$

The carrier set is the set of integers:

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

The Cayley table is an infinitely large table (since there are infinitely many integers, and we can add any two of them). Here is part of it:

$+$	$\ldots$	-1	0	1	2	3	$\ldots$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$
-1	$\ldots$	-2	-1	0	1	2	$\ldots$
0	$\ldots$	-1	0	1	2	3	$\ldots$
1	$\ldots$	-1	1	2	3	4	$\ldots$
2	$\ldots$	1	2	3	4	5	$\ldots$
3	$\ldots$	2	3	4	5	6	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

This structure is **associative**, so it qualifies as a **semigroup**.

Remark 43.3. This structure is also **commutative** (**abelian**), since the following is always true in this structure:

$$x + y = y + x$$

For instance, "$4 + 3 = 3 + 4$," "$5 + 2 = 2 + 5$," and so on for any other two integers we might add together.

Example 43.5. Recall the example of integer subtraction from Example 43.2:

$$\mathbf{Z} = (\mathbb{Z}, -)$$

Such a structure is **not associative**. For example:

$$3 - (4 - 2) \neq (3 - 4) - 2$$

Since it is not associative, this structure does not qualify as a semigroup. It is a groupoid, but it is *not* a semigroup.

43.3 Monoids

BEYOND SEMIGROUPS, there are monoids. A **monoid** is a semigroup with an identity (unit) element. So, a monoid is a structure comprised of a carrier set equipped with one binary operation, and that binary operation is associative, and it has an identity element.

Definition 43.3 (Monoid)**.** For any algebraic structure $\mathbf{S} = (A, \star)$ comprised of a carrier set A and a single binary operation $\star$, if $\star$ is associative and it has an identity (unit) element, then we will say that $\mathbf{S}$ is a **monoid**.

Terminology. A **monoid** is comprised of a carrier set and a single binary operation which (1) is associative, and (2) has an identity (unit) element.

Monoids are also groupoids and semigroups, since they satisfy the definitions for those types of algebras too. We can call a monoid an **associative groupoid with an identity**, or we can call it a **semigroup with an identity**.

Example 43.6. Suppose we have an algebraic structure $\mathbf{S} = (A, \star)$ where the set A is defined like this:

$$A = \{\mathsf{on}, \mathsf{off}\}$$

and the Cayley table is defined like this:

$\star$	on	off
on	on	off
off	off	on

This is associative. For instance, you can check that these all hold:

$$(\mathsf{on} \star \mathsf{off}) \star \mathsf{off} = \mathsf{on} \star (\mathsf{off} \star \mathsf{off}) \qquad (\mathsf{off} \star \mathsf{off}) \star \mathsf{on} = \mathsf{off} \star (\mathsf{off} \star \mathsf{on})$$
$$(\mathsf{on} \star \mathsf{on}) \star \mathsf{off} = \mathsf{on} \star (\mathsf{on} \star \mathsf{off})$$

It also has an identity element. It's "on." We can see it in the Cayley table:

$\star$	on	off
on	on	off
off	off	on

Example 43.7. Consider the integers $\mathbb{N}$ and the binary operation of addition (i.e., "+"). Together, these make up an algebraic structure:

$$\mathbf{N} = (\mathbb{N}, +)$$

The carrier set is the natural numbers:

$$\mathbb{N} = \{0, 1, 2, \ldots\}$$

The Cayley table is an infinitely large table (since there are infinitely many natural numbers, and we can add any two of them). Here is the start of it:

$+$	0	1	2	3	...
0	0	1	2	3	...
1	1	2	3	4	...
2	2	3	4	1	...
3	3	4	5	6	...
⋮	⋮	⋮	⋮	⋮	⋱

This is a monoid. It is **associative**, since it doesn't matter where we put the parentheses, e.g.:

$$4 + (3 + 2) = (4 + 3) + 2 \qquad 15 + (1 + 9) = (15 + 1) + 9$$

It also has an **identity** (**unit**) element, namely "0," since adding 0 to another number has no effect. For example:

$$7 + 0 = 7 \qquad 0 + 7 = 7$$

Since $\mathbf{N} = (\mathbb{N}, +)$ is associative and it has an identity, it qualifies as a **monoid**.

Remark 43.4. Notice that $\mathbf{N} = (\mathbb{N}, +)$ has no inverses. There is no number "x" that I can add to "7" to get back the identity "0." If I had negative numbers at hand, I could add "-7" to "7" to get the identity "0." But we don't have any negative numbers in this structure. Negative numbers belong to the **integers**, and this structure is defined only over **natural numbers**.

Example 43.8. Recall the example of integer subtraction from Example 43.2:

$$\mathbf{Z} = (\mathbb{Z}, -)$$

We already saw that this structure is not associative, but it also has no **identity** (**unit**) element.

It is tempting to think that "0" is an identity element here, because I can take 0 away from any number, and that won't have any effect. For instance, in each of these cases, "$x - 0 = x$":

$$7 - 0 = 7 \qquad (-3) - 0 = (-3) \qquad 97 - 0 = 97$$

However, this is true only if "0" appears on the right side of the subtraction symbol. If we put it on the left side, is not true that "$0 - x = x$." For instance:

$$0 - 7 = -7 \qquad 0 - (-3) = 3 \qquad 0 - 97 = (-97)$$

And of course, if there is no identity element, then this structure cannot have inverses either.

Since $\mathbf{Z} = (\mathbb{Z}, -)$ fails not only to be associative but also it fails to have an identity (unit) element, it does *not* qualify as a monoid. It is a groupoid, but not a monoid.

Remark 43.5. According to our definition from Chapter 42, "0" would be the **identity** (**unit**) element only if both of these were true in the structure:

$$x - 0 = x \qquad 0 - x = x$$

But they are not both true in the structure. Only "$x - 0 = x$" holds in this structure. Technically, we might say that here, "0" is an identity (unit) element **on the right**, but not **on the left**.

Example 43.9. Consider the integers $\mathbb{Z}$ and the binary operation of multiplication (often written as "$\times$" or "$\cdot$" but we'll just use the dot "$\cdot$" here). Together, these make up an algebraic structure:

$$\mathbf{Z} = (\mathbb{Z}, \cdot)$$

The carrier set is the set of integers:

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

The Cayley table is an infinitely large table (since there are infinitely many integers, and we can multiply any two of them). Here is part of it:

$\cdot$	$\dots$	-1	0	1	2	3	$\dots$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$
-1	$\dots$	1	0	-1	-2	-3	$\dots$
0	$\dots$	0	0	0	0	0	$\dots$
1	$\dots$	-1	0	1	2	3	$\dots$
2	$\dots$	-2	0	2	4	6	$\dots$
3	$\dots$	-3	0	3	6	9	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Remark 43.6. Notice that $\mathbf{Z} = (\mathbb{Z}, \cdot)$ does *not* have **inverses**. There is no number x I can multiply "5" by to get the identity "1." If we were dealing with fractions, then I could multiply "5" by "$\frac{1}{5}$" to get the identity "1." But we don't have any fractions in this structure. Fractions are **rational numbers**, and this structure is defined only over **integers**.

This structure is **associative**, since it never matters where we put the parentheses. For example:

$$2 \cdot (5 \cdot 3) = (2 \cdot 5) \cdot 3$$

It also has an **identity** (**unit**) element, namely "1," since multiplying any number by "1" has no effect. For instance:

$$7 \cdot 1 = 7 \qquad\qquad 1 \cdot 7 = 7$$

Since $\mathbf{Z} = (\mathbb{Z}, \cdot)$ is associative and it has an identity (unit) element, it qualifies as a **monoid**.

43.4 Summary

IN THIS CHAPTER, we looked at three different **types** of algebraic structures: groupoids, semigroups, and monoids.

- A **groupoid** is the simplest algebraic structure. It consists of a carrier set equipped with a single binary operation.
- A **semigroup** is a groupoid that is **associative**.
- A **monoid** is a semigroup that has an **identity** (unit) element.

[44]

GROUPS

In Chapter 43, we looked at **groupoids**, **semigroups**, and **monoids**. In this chapter, we will look at another type of algebraic structure, called a **group**.

44.1 Definition

Beyond groupoids, semigroups, and monoids, there are groups. A **group** is a monoid with inverses. So, a group is a structure comprised of a carrier set equipped with one binary operation, and that binary operation is (1) associative, (2) it has an identity element, and (3) it has inverses.

Terminology. A **group** is comprised of a carrier set equipped with a binary operation that is associative, has an identity (unit) element, and has inverses.

Definition 44.1 (Group)**.** For any algebraic structure $\mathbf{S} = (A, \star)$ comprised of a carrier set A and a single binary operation $\star$, if $\star$ is (1) associative, (2) has an identity (unit) element, and (3) has inverses, then we will say that $\mathbf{S}$ is a **group**.

Remark 44.1. Groups are *extremely* common, not just in mathematics, but also in nature. They seem to permeate everything.

The study of groups is called **group theory**, and it is a large, beautiful, and deep subject in and of itself. In the rest of this chapter, we will look at examples.

44.2 Two Isomorphic Groups

RECALL THE MUSEUM EXAMPLE from Chapter 39. We have $\mathbf{S} = (A, \star)$, with $A = \{m_0, m_1, m_2, m_3\}$ and this Cayley table:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

Remark 44.2. With the museum example, it doesn't matter where we put the parentheses (it is associative). For instance:

$$m_0 \star (m_1 \star m_2) = (m_0 \star m_1) \star m_2$$
$$m_0 \star m_3 = m_1 \star m_2$$
$$m_3 = m_3$$

The identity element is m_0. For example:

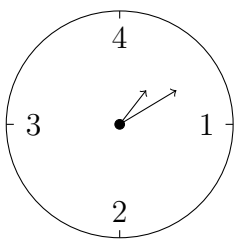
$$m_0 \star m_2 = m_2 \qquad m_2 \star m_0 = m_2$$

And each element has an inverse:

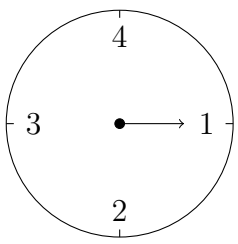
$$m_0^{-1} = m_0 \qquad m_1^{-1} = m_3$$
$$m_2^{-1} = m_2 \qquad m_3^{-1} = m_1$$

This is a **group**. It is associative, it has an identity element, and it has inverses.

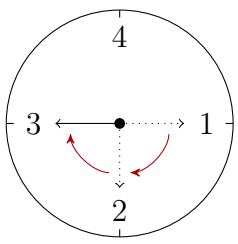
Now let's look at another, somewhat different example. Suppose we visit a distant planet whose clocks have only 4 hours on them, instead of 12. Like this:



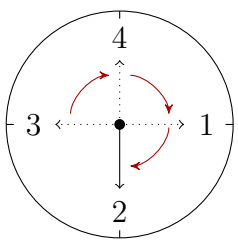
Suppose that it is 1 o'clock (let's just show the hour hand):



In 2 hours, what time will it be? The answer is this: it will be 3 o'clock. We can see this by moving the hour hand around two more ticks:



Let's suppose now that it is 3 o'clock. What time will it be after 3 more hours pass? It will be 2 o'clock. We can see this by moving the hour hand around three more ticks:



Remark 44.3. In this scenario, the set of numbers we have to work with are the numbers that are written on the face of the clock. So the **set** we have at hand is this:

$$A = \{4, 1, 2, 3\}$$

Remark 44.4. We are looking at **clock arithmetic** or **clock counting** here. When we get up to "4," we don't keep counting with "5." No, we wrap around and go back to "1." It's the same on our 12-hour Earth clocks. When we count up to "12," we wrap around and go back to "1" again.

Let's use our "$\star$" to talk about adding hours on this clock. Let's say that "$x \star y$" means that we add x and y on this 4-hour clock. So, for instance, we've seen that these are both true on this clock:

$$1 \star 2 = 3 \qquad 3 \star 3 = 2$$

Let's build a Cayley table for all of this clock arithmetic. Here it is, in full:

$\star$	4	1	2	3
4	4	1	2	3
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2

This is a **group**. It is associative, it has an identity element, and it has inverses.

Now, clock arithmetic and our museum example may not seem to have much in common, but in fact, these two algebraic structures are completely **isomorphic**. To prove this, we need to construct an **isomorphism**, i.e., a function f that preserves the binary operation. Here are our two carrier sets. The museum moves are on the left, and the numbers on the 4-hour clock are on the right:

Let's pick this mapping:

Remark 44.5. In this structure, it doesn't matter where we put the parentheses (it is associative). For instance:

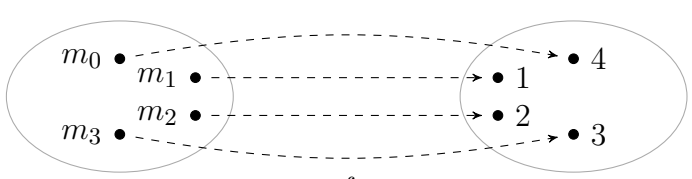
$$3 \star (4 \star 1) = (3 \star 4) \star 1$$
$$3 \star 1 = 3 \star 1$$
$$4 = 4$$

The identity element is 4. For example:

$$4 \star 2 = 2 \qquad 2 \star 4 = 2$$

And each element has an inverse:

$$4^{-1} = 4 \qquad 1^{-1} = 3$$
$$2^{-1} = 2 \qquad 3^{-1} = 1$$



f

Does this preserve the binary operation? To check, we need to make sure that corresponding elements appear in corresponding positions in the Cayley tables. Let's check m_0 and 4 first. We can see that they do appear in corresponding locations:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

$\star$	4	1	2	3
4	4	1	2	3
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2

We can also see that m_1 and 1 correspond:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

$\star$	4	1	2	3
4	4	1	2	3
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2

Ponder. Turn back to Chapter 39 and look at the pictures of movements through the museum galleries. Then look at the pictures above of the movements of the hands on our 4-hour clock. How are they similar? If you look past the surface details, and look at the deeper structure of these two examples, can you explain why it is that these two Cayley tables line up with each other so perfectly?

And m_2 and 2 do too:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

$\star$	4	1	2	3
4	4	1	2	3
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2

Finally, m_3 and 3 also correspond:

$\star$	m_0	m_1	m_2	m_3
m_0	m_0	m_1	m_2	m_3
m_1	m_1	m_2	m_3	m_0
m_2	m_2	m_3	m_0	m_1
m_3	m_3	m_0	m_1	m_2

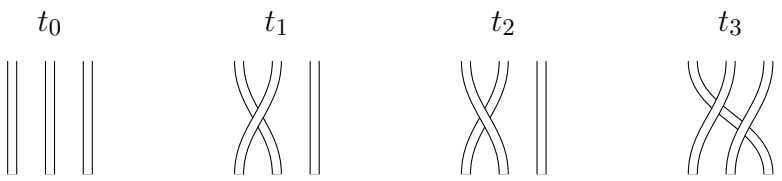
$\star$	4	1	2	3
4	4	1	2	3
1	1	2	3	4
2	2	3	4	1
3	3	4	1	2

So, the algebraic structures from our museum example and the 4-hour clock are **isomorphic**. Underneath the hood, they are exactly the same, and they differ only in the names of their elements.

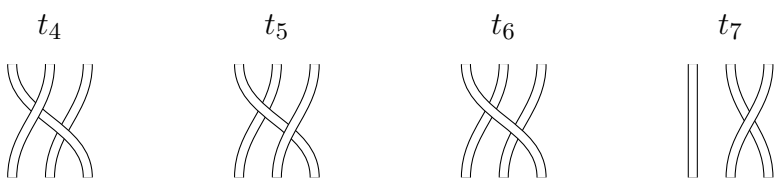
44.3 Braids

THERE ARE GROUPS IN KNOT THEORY. Suppose we have three ropes. There are different twists we can make, when we braid them. Here are a few different twists that are available to us:

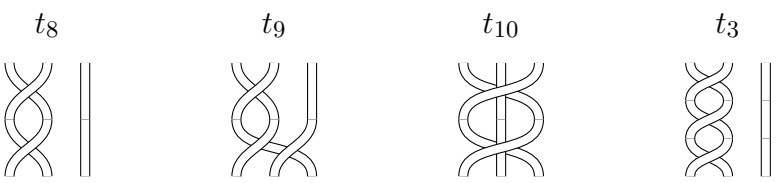
Remark 44.6. Notice that it matters whether a rope crosses in the front or behind another rope. For instance, in t_1, the center rope crosses over the top of the left rope, while in t_2, the center rope passes behind the left rope. These are different twists.



Here are some more twists we can make:



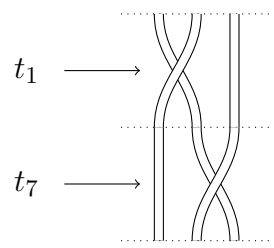
We can also do more than one twist at once. For instance, we can twist the ropes twice, three times, and so on:



So, we have a large set of twists that are available to us:

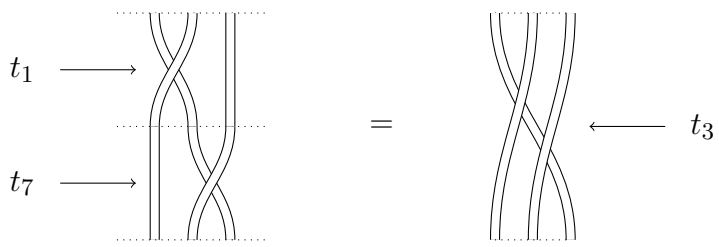
$$A = \{t_0, t_1, t_2, t_3, \ldots\}$$

We can combine these twists, by doing one after the other. For instance, we can do t_1, then t_7:



Remark 44.7. To combine t_1, we first do twist t_1 (i.e., we take the center rope and cross it over the top of the left rope), then we do twist t_7 (i.e., we take the rope on the right side, and cross it over the top of the center rope).

Notice that doing t_1 followed by t_7 (i.e., $t_1 \star t_7$) turns out to be just the same as doing t_3 all by itself:

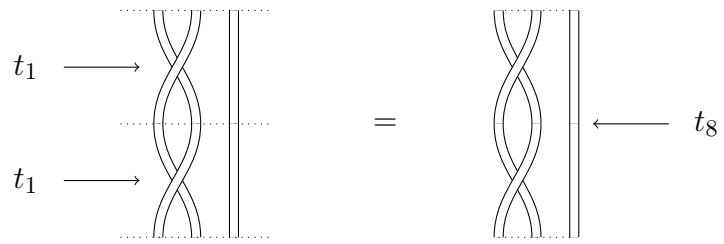


Hence, we can say that "$t_1 \star t_7$" is equal to "t_3":

$$t_1 \star t_7 = t_3$$

Remark 44.8. In this context, we can read "$t_1 \star t_7 = t_3$" out loud like this: "if we do twist t_1 then t_7, that's the same as just doing twist t_3."

Similarly, I can do t_1 followed by another t_1, and that's the same as doing t_8:



Hence:

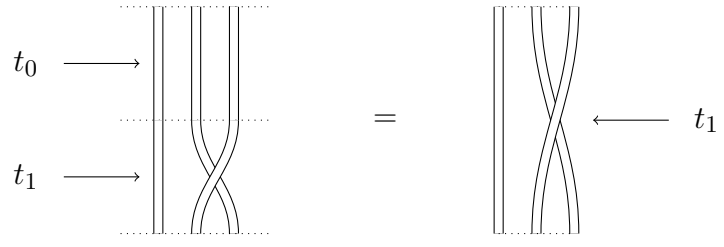
$$t_1 \star t_1 = t_8$$

In fact, we can combine any two twisting moves, and it turns out that it will be the same as just doing one of the other twisting moves.

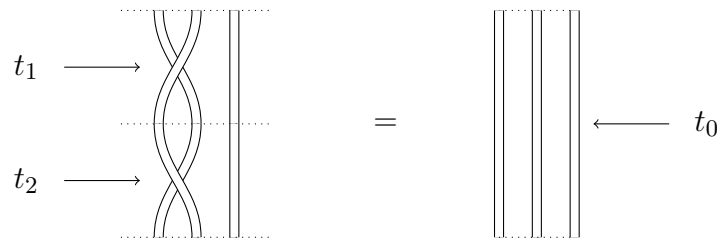
We can put our set of twisting moves and our combining operation together, to form an algebraic structure $\mathbf{T} = (A, \star)$. And this structure is a **group**.

First, it is associative, which is clear if you think about it for a bit. It doesn't matter which order I do the twists in. They all connect up end-to-end the same way.

Second, there is an identity element. It is t_0, since that has no effect on other twists. For instance, doing no twists (t_0) then t_1 is just the same as doing t_1 all by itself:



Finally, this structure has inverses, because for any twist I can make, there is another twist that can undo it. For instance, I can do t_1 followed by t_2, which is the same as t_0:



44.4 Arithmetic

WE FIND VARIOUS GROUPS IN ARITHMETIC. As a first example, take the integers $\mathbb{Z}$ and the binary operation of addition (i.e., "$+$"). Together, these make up an algebraic structure:

$$\mathbf{Z} = (\mathbb{Z}, +)$$

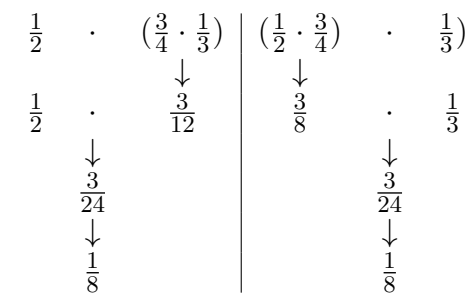
Remark 44.9. Recall from Chapter 43 that the structure $\mathbf{N} = (\mathbb{N}, +)$ does *not* have inverses, because $\mathbb{N}$ does not contain any negative numbers. Hence, $\mathbf{N} = (\mathbb{N}, +)$ is a **monoid**, but not a **group**. By contrast, $\mathbf{Z} = (\mathbb{Z}, +)$ does have negative numbers, so it has inverses, and hence it is a group.

This is a **group**. It is associative, there is an identity element (namely, "0"), and there are inverses (the inverse of every number is its positive/negative counterpart).

For a second example, take the set of fractions, i.e., the rational numbers $\mathbb{Q}$, and also the binary operation of multiplication (i.e., the symbol that can be written as "$\times$" or "$\cdot$," but let's just use "$\cdot$" for now). Together, these make up an algebraic structure:

$$\mathbf{Q} = (\mathbb{Q}, \cdot)$$

This is a **group** as well. It is associative. For instance:



Remark 44.10. Recall from Chapter 43 that the structure $\mathbf{Z} = (\mathbb{Z}, \cdot)$ does *not* have inverses, because there is no number in $\mathbb{Z}$ that you can multiply by 5 that will cancel it out and give us back 1. We would need the fraction $\frac{1}{5}$ to cancel out 5, but there are no fractions in $\mathbb{Z}$. So, $\mathbf{Z} = (\mathbb{Z}, \cdot)$ has no inverses, and hence it is a **monoid**, but not a **group**. By contrast, $\mathbf{Q} = (\mathbb{Q}, \cdot)$ does have the fractions we need to cancel each other out, so it has inverses, and hence it *is* a group.

There is an identity element, namely "$\frac{1}{1}$" because multiplying any fraction by $\frac{1}{1}$ has no effect. For instance:

$$\frac{3}{8} \cdot \frac{1}{1} = \frac{3}{8} \qquad \frac{1}{1} \cdot \frac{3}{8} = \frac{3}{8}$$

Finally, there are inverses. The inverse of any fraction is the same fraction, but turned upside down. They cancel each other out. For instance:

$$\frac{3}{8} \cdot \frac{8}{3} = \frac{24}{24} = \frac{1}{1} \qquad \frac{1}{4} \cdot \frac{4}{1} = \frac{4}{4} = \frac{1}{1} \qquad \frac{33}{18} \cdot \frac{18}{33} = \frac{594}{594} = \frac{1}{1}$$

44.5 Summary

IN THIS CHAPTER, we looked at the definition of groups, and we looked at some example of groups.

- A **group** is an algebraic structure comprised of a carrier set equipped with a single binary operation, and (1) it is associative, (2) it has an identity (unit) element, and (3) it has inverses.
- To illustrate the wide variety of groups that appear in math and nature, we looked at a variety of examples ranging from clocks, to braids, to shapes, to arithmetic.

[45]

FURTHER READING

To pursue algebras further, the following list may offer some helpful starting points. Much of the literature on algebra is focused on **groups**, but there are more complex types of algebraic structures called **rings** and **fields**, which also receive a lot of treatment. **Linear algebra** is another big sub-discipline of algebra. After getting comfortable with groups, one can easily turn their sites on linear algebra, rings, and fields.

For more introductory level discussions that explain concepts rather than theorems and proofs, the following may be helpful:

- Sawyer (2007, chs. 5–7) offers a beginner level discussion of basic arithmetic and algebraic structures.
- Sawyer (1982, chs. 3–4, 7, 8, and 14) provides simple explanations of many aspects of abstract algebra. See chap-

ters 3–4 and 7 for the development of abstract algebraic structures generally, chapter 14 for groups, and chapter 8 for linear algebra.

- Sawyer (2018) offers one of the best discussions of rings and fields I know of. The discussion is aimed at explaining the basic intuitions behind how these structures lead to the kinds of calculations many learn in "algebra" at school.
- Stewart (1995, chs. 6, 7, and 15) offers excellent introductory discussion of groups (chapter 7), rings and fields (chapter 6), and linear algebra (chapter 15).
- Stewart and Tall (2015, chs. 13 and 10) offers a somewhat introductory discussion of groups (in chapter 13) and fields (chapter 10).

For accessible but more thorough discussions, these may be helpful:

- Dos Reis and Dos Reis (2017) is the best beginner's book for algebra I know of. It is thorough, and slow paced, and explains most things very well. In addition, the answer to every exercise is explained fully. To learn algebra, this is a great first place to start.
- Pinter (2010) is an accessible introduction to algebraic structures. It can be quite useful as a companion to read alongside Dos Reis and Dos Reis (2017) and Baumslag and Chandler (1968). There is one problem with the text: the exercises are full of typos and errors. This is too bad, because the exercises are often very interesting. That being said, if you want to do the exercises, answers to almost all of them can be found online if you look hard enough, and that will help you straighten out any errors that you can't straighten out yourself.

- Baumslag and Chandler (1968) is another great text to start with, or to use as a second text after Dos Reis and Dos Reis (2017). The notation is quite different from what most other books use, but the authors do explain their notation, so this problem is easily overcome. The nice thing about this text (and all Schaum's Outlines books) is that all exercises have solutions.
- Saracino (2008) is a good textbook on abstract algebra. If you can find solutions online, this is a great text to work through after covering Dos Reis and Dos Reis (2017) or Baumslag and Chandler (1968).
- Schmidt (1966) is an old, out-of-date text, but in many ways it is simple and can be quite accessible.
- Warner (1990) is an advanced text that covers a great many aspects of modern algebra.

Part [9]

EQUIVALENCE CLASSES

When we work with collections of objects, we often want to partition the collection up into disjoint sub-collections. And often, when we do this, our goal is to put all "equivalent" items into the same buckets. For instance, maybe we want to sort all of the employees of a company into different buckets by their skillsets. The great advantage of doing this is that, for certain tasks, all we really care about are the buckets, instead of the individuals inside the bucket. For next week's project, I don't care who it is, I just need someone with the right skills to help.

In Part 9, we look first at how to **partition** a set into **disjoint** subsets. Then we turn to the idea of partitioning a set into disjoint **equivalence classes**. Equivalence classes turn up in quite a lot of places in the world of math.

[46]

PARTITIONS

IN CHAPTER CHAPTER 22, we looked at the concept of **structures**. A structure is one or more sets (called the **base sets** or **carrier sets**), equipped with any number of functions or relations that add some structure to the base sets. In this and the following chapters, we will look at a few structures that consist of a **base set** and a **single relation**. The relations we will look at are special, because each of them arranges the items of the base set into a particular configuration.

The first such structure we will discuss are called **partitions**, and that is the topic for this chapter. A **partition** of a set A cleanly cuts A up into distinct, non-overlapping subsets. If we glue those subsets back together, we end up with the original set A again.

In the next chapter, we will look at **equivalence classes**, which arise by equipping a set A with an **equivalence rela-**

Remark 46.1. We will look at three special kinds of structures: (i) a set equipped with an **equivalence relation**, (ii) a set equipped with an **ordering relation**, and (iii) a set equipped with a **lattice** relation.

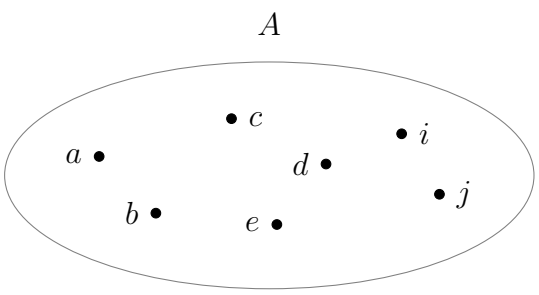
tion. An equivalence relation partitions a set into groups of equivalent items.

Following that, we will look at **ordered sets**, which are sets equipped with an **ordering relation** (it puts some or all of the items from the base set into some ordering). Then in the following chapter, we will look at **lattices**, which are sets equipped with a very special kind of ordering relation that gives the whole set the shape of a lattice.

46.1 Partitions

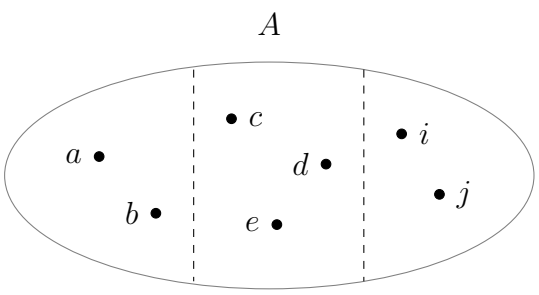
Terminology. A **partition** of a set A is a collection of subsets of A, which do not overlap each other, but when glued together, they make up the whole set A.

ONE WAY TO BREAK UP A SET is to **partition** it, i.e., to cut it up into distinct, non-overlapping pieces. For example, take a set A:



Then cut it up into, say, three non-overlapping pieces:

Remark 46.2. The dashed lines indicate places where we divide the set A up into subsets. These two dashed lines indicate that we have broken up A into three subsets: one that contains a and b, another that contains c, d, and e, and a third that contains i and j.



These three pieces make up a partition of A. Basically, a **partition** of A is a collection of subsets of A, none of which overlap each other, but if you glue them back together, you get the whole subset A again. We can call each "piece" of the partition (i.e., each subset) a **cell** of the partition.

In this case, we started with a set A:

$$A = \{a, b, c, d, e, i, j\}$$

Then we broke it up into three **cells** (i.e., three subsets). Let's call them S_1, S_2, and S_3:

$$S_1 = \{a, b\} \qquad S_2 = \{c, d, e\} \qquad S_3 = \{i, j\}$$

These three cells do not overlap, in the sense that they share no elements. There is no element that belongs in more than one of these cells. Each element belongs to exactly one, and only one, cell.

Also, if we glue all of these cells back together, we end up with the original set. Hence, if we combine S_1 and S_2, we end up with $\{a, b, c, d, e\}$, and then if we combine that with S_3, we end up with the original set $\{a, b, c, d, e, i, j\}$ again.

Remark 46.3. The three cells S_1, S_2, and S_3 form a **partition** of A, because they do not overlap (they have no elements in common), and when you glue them all back together, you get A again.

46.2 Definition

We can use set-theoretic language to define what a partition is more exactly. We can use set *intersection* to define what it means for the cells to be non-overlapping. Remember that the **intersection** of two sets is the set of items they have in common.

In this case, none of the cells have any items in common. If we compare S_1 with S_2, we can see that they have nothing in common, so their intersection is empty. Likewise, if we

Terminology. Recall from Section 9.2 that the **intersection** of two sets A and B, which we denote $A \cap B$, is the set of elements that exist in both A and B. If the intersection is **empty**, which we denote $A \cap B = \varnothing$, then that means A and B share no elements.

compare S_2 with S_3, we can see that they have no elements in common, so their intersection is empty too. And the same goes if we compare S_1 and S_3. So, we can say that the intersection of any two of our cells is **empty**:

$$S_1 \cap S_2 = \varnothing \qquad S_2 \cap S_3 = \varnothing \qquad S_1 \cap S_3 = \varnothing$$

Terminology. If the intersection of two sets is empty, we say they are **disjoint**. If a number of sets are disjoint from each other, we say they are **pairwise disjoint** (because we can pair them up and see that each is disjoint from each).

We say that these subsets are **pairwise disjoint**. By saying they are *disjoint*, we mean to say that they have no elements in common (their intersection is empty). By saying they are *pairwise* disjoint, we mean that *each pair* of them is disjoint: this one is disjoint from that one, and this one is disjoint from that other one, and so on. Let's put this down in a definition:

Definition 46.1 (Pairwise disjoint)**.** For any collection C of sets S_1, S_2, ..., we will say that the sets in C are **pairwise disjoint** if, for each pair of sets S_n and S_m in C, $S_n \cap S_m = \varnothing$.

We can use the concept of set *union* to define what it means for the cells to make up the whole set A again if you glue them back together. Recall that the **union** of two sets is all of the elements from both sets combined. So, to "union" two sets, is to combine them.

Terminology. Recall from Section 9.1 that the **union** of two sets A and B, which we denote $A \cup B$, is the set we get when we combine all of the elements from A and all of the elements from B.

Hence, when we say that if you glue our cells S_1, S_2, and S_3 back together you get the whole set A again, we can instead say this: the *union* of S_1, S_2, and S_3 is equal to the original set A:

$$S_1 \cup S_2 \cup S_3 = A$$

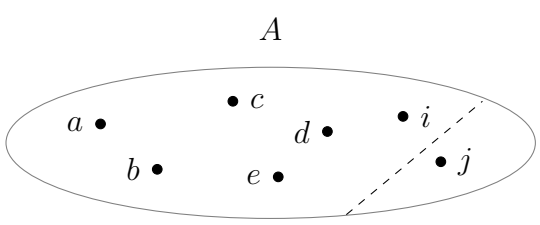
With all of that at hand, we can now define a partition using these concepts from set theory. We can say that a partition of A is a collection of subsets of A which (a) are pairwise dis-

joint, and (b) the union of which is equal to A. Let's put that in a definition:

Definition 46.2 (Partition)**.** For any set A and any collection C of subsets of A, we will say that C is a **partition** of A if (i) all of the subsets in C are pairwise disjoint, and (ii) the union of all subsets in C is equal to A. We will call each subset of a partition C a **cell** of C.

46.3 Examples

Example 46.1. Here is a partition:

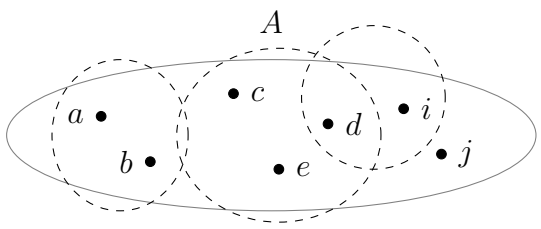


Where the cells are these:

$$S_1 = \{a, b, c, d, e, i\} \qquad S_2 = \{j\}$$

These subsets are pairwise disjoint (no element is shared), and when glued together, they make up A (the union of S_1 and S_2 equals A).

Example 46.2. Suppose we take the following subsets of A (marked by circles):



So that we end up with these subsets:

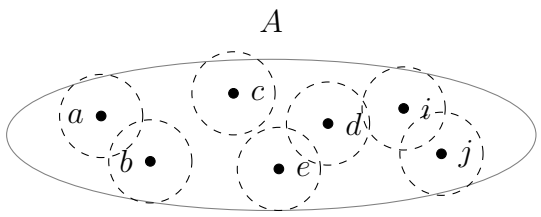
$$S_1 = \{a, b\} \qquad S_2 = \{c, d, e\} \qquad S_3 = \{d, i\}$$

This is *not* a partition on A, for two reasons. First, the subsets are not pairwise disjoint. In particular, S_2 and S_3 have an element in common, namely d. So $S_2 \cap S_3 \neq \varnothing$ (i.e., the intersection of S_2 and S_3 is not empty). On the contrary, $S_2 \cap S_3 = \{d\}$.

Second, the union of these subsets does not add up to A. If we union S_1 and S_2, we get $\{a, b, c, d, e\}$, and if we union that with S_3, we get $\{a, b, c, d, e, i\}$. But that misses out on j, which is an element in A. Hence, we can see that if we glue these pieces back together, we don't get back the full set A.

Remark 46.4. The dashed circles in the picture indicate the subsets. Points that appear inside a dashed circle are elements of that subset. If two dashed circles overlap a point, then that point belongs to both circles (i.e., it belongs to both subsets).

Example 46.3. Consider the following subsets of A:



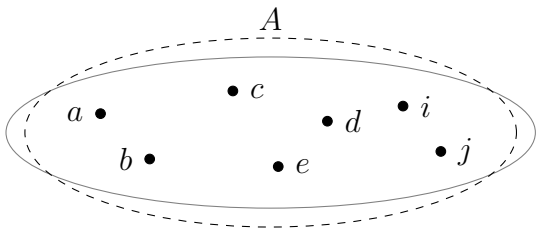
We have these cells:

$$S_1 = \{a\} \qquad S_2 = \{b\} \qquad S_3 = \{c\} \qquad S_4 = \{d\}$$
$$S_5 = \{e\} \qquad S_6 = \{i\} \qquad S_7 = \{j\}$$

Remark 46.5. In this example, every point has its own dashed circle drawn around it. So every point from A is put into its own cell. We can think of this as a limit case of partitions: it is the case where we partition the set A into the **smallest** possible pieces: one cell for each point!

This is a partition of A, because it breaks it up into 7 pairwise disjoint pieces, the union of which equals A.

Example 46.4. Consider the following subsets of A (there is one subset, marked by a circle):



We have this one cell:

$$S_1 = \{a, b, c, d, e, i, j\}$$

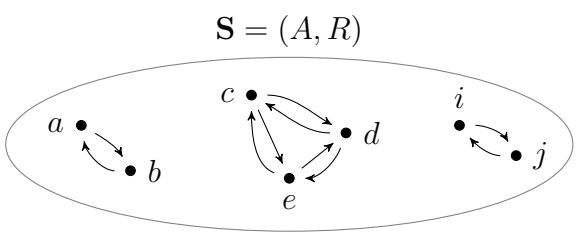
This is a partition of A, because it breaks the original set up into a single piece. None of the subsets overlap (because there's only one subset), and if we combine the subsets, they "add up" to the original set A (indeed, the one subset just is the whole set A). So this is a 1-piece partition (like a 1-piece jigsaw puzzle). It's a partition, but only in a **vacuous** sense.

Remark 46.6. In this example, all of the points belong in one dashed circle. So every point from A is put into the same cell. We can think of this as the opposite limit case of partitions: it is the case where we partition the set A into the **biggest** possible piece: all of the points go in the same big piece.

Terminology. Recall from Section 8.3 that something satisfies a definition **vacuously** if there is nothing to do to satisfy the definition.

46.4 Relations

SOME RELATIONS PARTITION A SET. For example, suppose we have a structure $\mathbf{S} = (A, R)$ comprised of a set A and a relation R, which looks something like this:



The base set A is this:

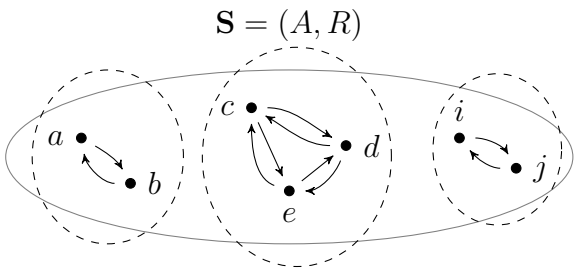
$$A = \{a, b, c, d, e, i, j\}$$

And the relation R is this:

$$\begin{aligned} R = \{&(a, b), (b, a), \\ &(c, d), (d, c), (d, e), (e, d), (c, e), (e, c), \\ &(i, j), (i, j)\} \end{aligned}$$

Notice that this relation connects the elements in such a way that they clump together into separate groups. We can draw circles around each distinct clump:

Remark 46.7. The relation connects the elements together into distinct clumps. We can draw circles around the clumps.



If we make subsets from these circles, we get:

$$S_1 = \{a, b\} \qquad S_2 = \{c, d, e\} \qquad S_3 = \{i, j\}$$

This collection of subsets is a **partition** of A, which is obvious just by looking at the picture: none of these subsets

Remark 46.8. We can check that this collection of subsets truly qualifies as a partition. (1) Are these subsets pairwise disjoint? Yes, because none of them share any elements. (2) When we union all of them together, do we end up with our original set A? Yes, if we combine these subsets, we get A. So yes, this is a genuine partition of A.

overlap, and if we glued them together, we'd end up with our set A.

Because this partition arises from the relation R, we say that the relation **induces** the partition, or we say that the partition **is induced** by the relation R.

However you want to say it, the basic idea is just that the relation itself breaks the set A up into distinct, non-overlapping clumps. The relation **partitions** the set. Hence, we can refer to the whole structure $\mathbf{S} = (A, R)$ as a **partitioned structure**, because it is a set A which is partitioned into distinct cells by the relation R.

Terminology. Given a structure $\mathbf{S} = (A, R)$, if R partitions A, we say that R **induces** the partition. We can call the whole structure a **partitioned structure**, or just "a partition" for short.

46.5 Summary

IN THIS CHAPTER, we learned about partitions and partition structures.

- A **partition** of a set A is a collection of subsets of A which are pairwise disjoint, and which when unioned are equal to A. The subsets are called the **cells** of the partition.
- If a relation R on a set A groups the elements of A into non-overlapping clumps which can be taken as the cells of a partition of A, then we say that R **induces** that partition on A.
- A **partitioned structure** is a structure $\mathbf{S} = (A, R)$ where the set A is partitioned by the relation R (i.e., a partition of A is induced by R).

[47]

EQUIVALENCE RELATIONS

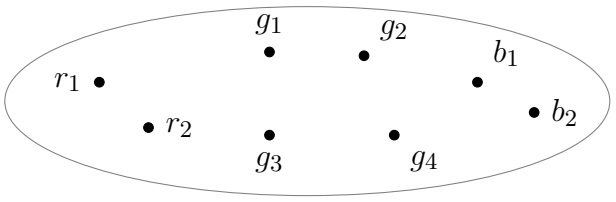
IN CHAPTER 46, we looked at **partitions** and also **partitioned structures**. A partition on a set A is a collection of cells (i.e., disjoint subsets of A) which are such that, if you glue them all back together, you end up with A again. In this chapter, we will look at a special kind of partitioned structure, which is based on the notion of **equivalence**.

47.1 Equivalence relations

SOME RELATIONS PAIR UP all the elements that are equivalent to each other in some way. We call these **equivalence relations**. Consider the following example. Suppose I have two red marbles (call them r_1 and r_2), three green marbles

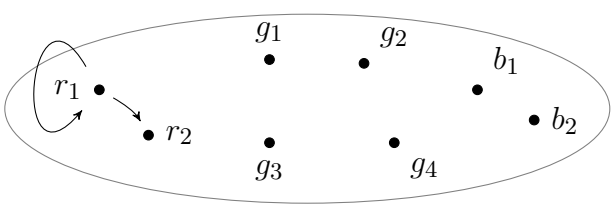
Terminology. An **equivalence relation** pairs up all elements that are equivalent to each other in some way.

(call them g_1, g_2, and g_3), and two blue marbles (call them b_1 and b_2):

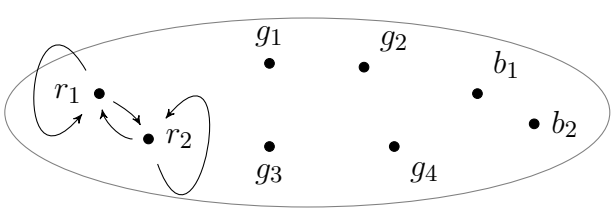


Now suppose we connect up each marble to every marble with the same color (including itself). So, for instance, let's start with r_1. Let's connect it to every red marble. Which marbles are red? It's r_2, and also itself, so we add those arrows:

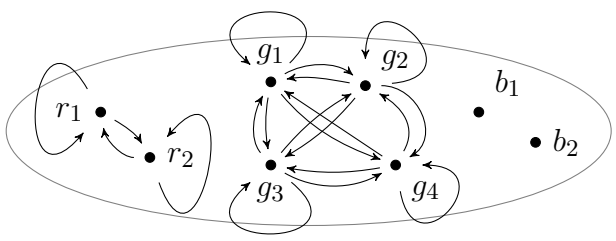
Remark 47.1. Remember: we are connecting r_1 to every red marble. When we say "every red marble," we mean *every* red marble, include r_1! If r_1 weren't connected to itself, then it wouldn't be connected to *every* red marble. There would be one red marble that it wouldn't be connected to, namely itself.



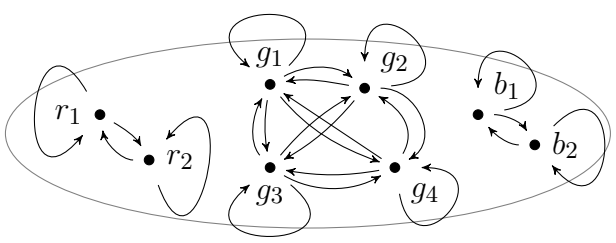
Now let's turn to r_2, and connect it to every red marble. There's r_1, and itself, so we draw in those arrows too:



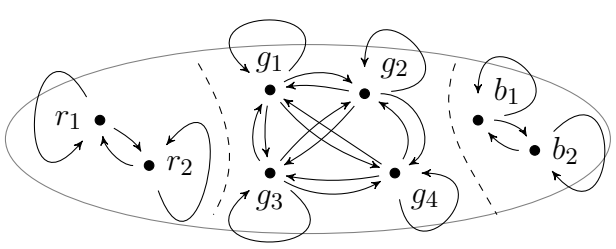
Next, we do the same for the green marbles, and connect each one to every green marble:



Finally, we connect each blue marble to every blue marble, which gives us this:



We can see from the picture that this relation *partitions* our original set into three cells:



Remark 47.2. Notice that *each* green marble is connected to *every* green marble, even itself. All of the green marbles are *thoroughly connected* to each other. Pick any two green marbles from the picture, and you'll find an arrow between them (even if you check whether there is an arrow from a green marble to itself). Likewise, all of the red marbles are thoroughly connected, and all of the blue marbles are thoroughly connected.

Remark 47.3. Each cell collects up the items that are equivalent. In the left cell we have all marbles that are red, in the middle cell we have all marbles that are green, and in the right cell we have all marbles that are blue.

The arrows tell us which items are **equivalent** (in color).

They show us the clump of items that are red, and the clump of items that are green, and the clump of items that are blue.

Let us write out the structure that we have here, in symbols. First, we have a base set. Let's call it A:

$$A = \{r_1, r_2, g_1, g_2, g_3, g_4, b_1, b_2\}$$

Then, we have a relation. Let's call it R. It is comprised of the following pairings:

$$\begin{aligned} R = \{&(r_1, r_1), (r_1, r_2), (r_2, r_1), (r_2, r_2),\\ &(g_1, g_1), (g_1, g_2), (g_1, g_3), (g_1, g_4),\\ &(g_2, g_1), (g_2, g_2), (g_2, g_3), (g_2, g_4),\\ &(g_3, g_1), (g_3, g_2), (g_3, g_3), (g_3, g_4),\\ &(g_4, g_1), (g_4, g_2), (g_4, g_3), (g_4, g_4),\\ &(b_1, b_1), (b_1, b_2), (b_2, b_1), (b_2, b_2)\} \end{aligned}$$

Remark 47.4. Our base set is the set of marbles, and our equivalence relation pairs up all equivalent items. Notice that the red pairings make up the **product** of $\{r_1, r_2\}$, the green pairings make up the **product** of $\{g_1, g_2, g_3, g_4\}$, and the blue pairings make up the **product** of $\{b_1, b_2\}$. In other words, each cell has every possible combination of pairings of the elements in the cell: every possible pairing of reds, every possible pairing of greens, and every possible pairing of blues.

Put these together, and we have a structure. Let's call it $\mathbf{S}$:

$$\mathbf{S} = (A, R)$$

We might call this an **equivalence structure**, because it is a set partitioned into clumps of equivalent items.

Terminology. An **equivalence structure** is a structure built by taking a base set and equipping it with an equivalence relation. The equivalence relation partitions the base set into cells of equivalent items.

47.2 Notation

EQUIVALENCE IS A FANCY WORD for similarity. When we say x and y are equivalent, we aren't saying that x and y are identical. We are saying they are "equal" only in some specific way. Hence, we use the word "equivalent."

In the example above, the relation makes explicit which objects are equivalent in color. Each arrow pointing from one

marble x to another marble y indicates that x is the same color as y. Mathematicians have a special symbol for equivalence. It looks like this:

$$\sim$$

When we're dealing with an equivalence relation, instead of writing R, we can write $\sim$ instead.

Hence, instead of describing our structure as "$\mathbf{S} = (A, R)$," we can describe it like this:

$$\mathbf{S} = (A, \sim)$$

When we use "R" as a name for a relation, we can assert that R connects x to y by writing this:

$$R(x, y)$$

When we're dealing with an equivalence relation, we can use the equivalence symbol instead, and write this:

$$\sim (x, y)$$

In fact, mathematicians usually write that like this:

$$x \sim y$$

Read it aloud as, "x is equivalent to y." This is just a different way of writing "$\sim (x, y)$," but writing it with the symbol in between x and y makes it look a little like this:

$$x = y$$

But "$x = y$" asserts that x and y are identical objects. What we want to say here instead is that x and y are "equal" only *in color*. That is why we can use the equivalence symbol instead.

Notation 47.1. To denote an **equivalence relation**, we will use the tilde character (namely, "$\sim$"). To denote an equivalence structure $\mathbf{S}$ built from a base set A and an equivalence relation $\sim$, we will write $\mathbf{S} = (A, \sim)$. To denote that the equivalence relation connects x to y, we will write $x \sim y$, which we can read aloud as, "x is equivalent to y."

Remark 47.5. When we put a relation or function symbol up front, before x and y, as when we write "$\sim (x, y)$" or "$R(x, y)$," we say that we write the symbol using **prefix** notation (because it *prefixes* or comes before x and y). When we put the symbol in between x and y, as when we write "$x \sim y$" or "xRy," we say that we write it using **infix** notation (because it is fixed *in between* x and y). We can do the same with, say, the addition symbol: we can write "$3 + 4$" (infix notation) or "$+(3, 4)$" (prefix notation). Whether we use prefix or infix notation, it doesn't really matter. Both are just two different ways of writing down the same thing.

47.3 The Definition

LET US PUT DOWN A DEFINITION for equivalence relations. Formally speaking, mathematicians characterize an equivalence relation as any relation that is **reflexive**, **symmetric**, and **transitive**. Hence, we can define equivalence relations as follows:

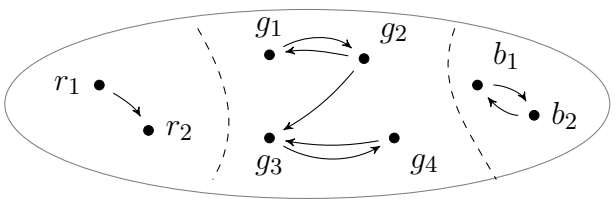
Definition 47.1 (Equivalence relation)**.** Given a relation R on a set A, we will say that R is an **equivalence relation** if it is reflexive, symmetric, and transitive. To denote an equivalence relation, we will use this symbol: $\sim$. To assert that x and y are equivalent under this relation, we will write this: $x \sim y$.

Terminology. Formally speaking, an **equivalence relation** is any relation that is reflexive, symmetric, and transitive. Recall from Chapter 20 that for any given relation R on a set A:

- R is **reflexive** if every point is connected to itself, i.e., $R(x, x)$ for every x in A.
- R is **symmetric** if every connection goes both ways, i.e., if $R(x, y)$ then $R(y, x)$.
- R is **transitive** if the start- and end-point of every two-arrow chain is connected, i.e., if $R(x, y)$ and $R(y, z)$ then $R(x, z)$.

Why do we define an equivalence relation as any relation that is reflexive, symmetric, and transitive? Well, think about our picture of the marbles above. What can we see there? We can see that the relation partitions the set into cells. But there's more. We can also see that it *thoroughly connects* all of the items inside a cell to each other.

There are multiple ways that a relation can partition a set. For instance, here is one:



Remark 47.6. Not every relation that partitions a set **thoroughly connects** all of the points inside each cell.

However, this does not thoroughly connect the points in each cell to each other. In these cells, there are some connections missing. For example, in the left cell, r_2 is not connected

back to r_1, or in the middle cell, g_1 is not connected to g_3. So this picture presents us with a partition, but it is not one where all of the points in each cell are thoroughly connected to each other.

By contrast, an equivalence relation *does* thoroughly connect all the points in a cell to each other. If you look at our earlier picture above, you will see that, in each cell, all of the points in each cell are connected to each other. There are no missing connections in each cell.

So how do we characterize this idea that the points in each cell are "thoroughly connected" to each other? It turns out that the idea is captured exactly by saying the relation is reflexive, symmetric, and transitive. Let's look at each of these in turn, to see why.

Remark 47.7. To say that a cell is *thoroughly connected* is to say that every point in the cell is connected to (i.e., has an arrow pointing to) every point in the cell, including itself.

47.3.1 Reflexivity

First of all, if *each point* is connected to *every point* (in the cell), then each point must have an arrow that points back **to itself**. To see this, suppose we only have two points in a cell:

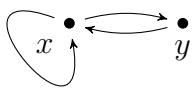
Let's look at x first. We want this point to be connected to every point in the cell. So, let's connect x to y:

Is x connected to every point in the cell now? The answer is no. There are two points in the cell, and x is connected to only one of them. It is connected to y, but not to x (itself). So, we need to connect it back to itself. Hence:

Remark 47.8. Is x connected to every point in the cell? If it's missing a connection to itself, then there's at least one point in the cell that it's *not* connected to, namely itself.

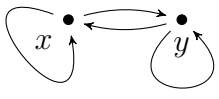
Now x is connected to every point in the cell. There are two points in the cell (namely, x and y), and x is connected to both of them.

Next, consider y. We also want y to be connected to every point in the cell. Let's connect y to x:



Remark 47.9. In order for all of the points to be connected, *each point* needs to be connected *each point* (including itself). So, this cell is fully connected only when x is connected to each point, *and* when y is connected to each point too.

Is y connected to every point now? The answer is no. There is one point it is *not* connected to, namely itself. Hence, we need to add an arrow looping back to y as well:



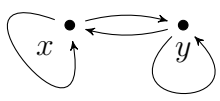
Now y is connected to every point in the cell. There are two points (namely, x and y), and y is connected to both of them.

This makes it clear that every point needs to be connected not just to the *other* points in the cell, it also needs to be connected to *itself*. So, if a relation is one that *thoroughly connects* every point in a cell, then it needs to be **reflexive**: it must connect each point to itself.

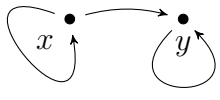
Remark 47.10. Recall from Chapter 20 that a relation R is **reflexive** if each point x is connected to itself, i.e., if $R(x, x)$ for every x.

47.3.2 Symmetric

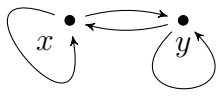
Second, if a point is connected to another point, then that other point must be connected back. Each connection needs to go **both ways**. To see this, consider our two-point cell again:



Suppose that y is connected to itself, but *not* x:



In this picture, we can see that x is connected to y, but y is not connected back to x. Well, is y connected to every point in the cell? The answer is no: it may be connected to itself, but it is not connected to x. So, if we want it to be connected to every point in the cell, we need to add that arrow back in:



We can see from this that, in order for these points to be *thoroughly connected*, if one point is connected to another, the reverse connection needs to be there too. Otherwise, there

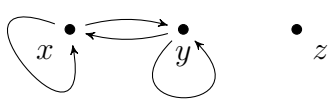
Remark 47.11. Connections do not automatically go both ways. Just because x is connected to y by a relation R does not necessarily mean y is connected back to x in that same way (Harry can love Selma, but Selma need not love Harry back). An arrow from x to y only connects x to y. If we want y to be connected back to x, we need a second arrow, going from y to x.

Remark 47.12. Recall from Chapter 20 that a relation R is **symmetric** when each connection goes both ways, i.e., if $R(x, y)$, then $R(y, x)$.

will be a missing connection. So, the relation needs to be **symmetric**.

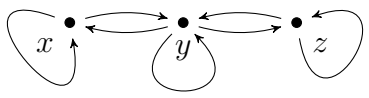
47.3.3 Transitivity

Third, chains need to be thoroughly connected. To see this, suppose we have a third point in our cell:

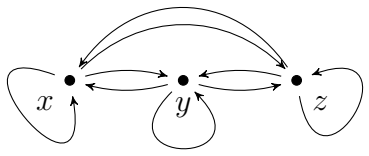


Remark 47.13. In this picture, there are two links in a chain: x and y are fully connected to each other (that's one link in the chain), and y and z are fully connected to each other (that's a second link in the chain). When we say they are "fully connected to each other," we mean they have all connections between them (both reflexive and symmetric arrows). However, the start- and end-points of this two-link chain are *not* connected. There are no arrows going from x to z, nor from z to x.

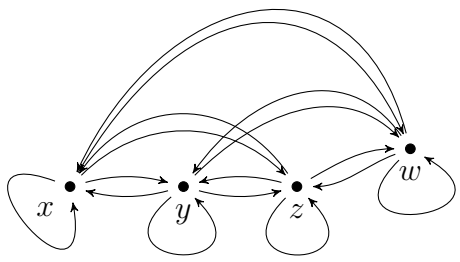
And suppose that y and z are fully connected too:



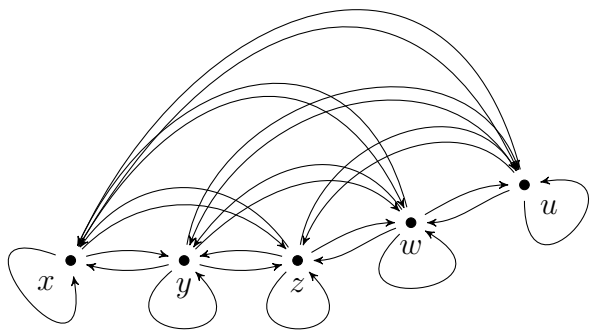
Now consider x. Is x connected to every point in the cell? The answer is no. It is not connected to z (and similarly, z is not connected to x). So, even though x is connected to y and y is connected to z in a chain, we are missing a connection between the start of that chain (namely, x) and the end of that chain (namely, z). So we need to add those arrows too:



If there were a fourth point in the cell, we'd need to add more arrows in this fashion:



And if there were a fifth point, we'd have to add even more:



This makes it clear that, in order for the points in the cell to be thoroughly connected, chains need to be thoroughly connected too. If x is connected to y and y is connected to z, then x must *also* be connected to z. Otherwise, there will be a missing connection. (And since the relation also needs to

Remark 47.14. Note that once we add w to the cell, we get more two-link chains. There is $x \mapsto y \mapsto z$ and $y \mapsto z \mapsto w$, but then there is also $x \mapsto y \mapsto w$, and $x \mapsto z \mapsto w$. And for each of these, the start- and end-points need to be connected. Likewise, when we add u to the cell, we get even more two-link chains (e.g., $x \mapsto z \mapsto u$, or $x \mapsto w \mapsto u$), all of which need their start- and end-points connected too.

Remark 47.15. Recall from Chapter 20 that a relation R is **transitive** when the start- and end-points of every two-arrow chain are connected, i.e., if $R(x, y)$ and $R(y, z)$, then $R(x, z)$.

be symmetric, z needs to be connected back to x, or else there will again be a missing connection.) And, this goes for *every* "two-links" in the chain. For every two links that are fully connected, the start- and end-points of those two links need to be connected to. So, the relation needs to be **transitive**.

Remark 47.16. If a relation thoroughly connects up all the points in a cell, it will be reflexive (no points will be missing a connection to themselves), it will be symmetric (no connections will be missing a return arrow), and it will be transitive (no chains will be missing connections).

The aforegoing shows why we can define an equivalence relation as any relation that is reflexive, symmetric, and transitive. Any relation that partitions a set into cells, and also thoroughly connects the points in each cell, is going to be reflexive, symmetric, and transitive. Otherwise, there would be some connections missing.

To say the relation must be reflexive guarantees that every point is connected to itself, to say it must be symmetric guarantees that every connection goes both ways, and to say it must be transitive guarantees that all the points in a chain are connected up too. This way, there simply aren't any missing connections inside a cell.

47.4 Summary

IN THIS CHAPTER, we learned about **equivalence relations**, which are relations that partition a set into cells, and which thoroughly connects all the points in each cell to each other.

- Formally speaking, an **equivalence relation** is any relation that is reflexive, symmetric, and transitive.
- We can use the "$\sim$" symbol to denote an equivalence relation, and to say that "x is equivalent to y," we can write this: "$x \sim y$."

[48]

EQUIVALENCE CLASSES

IN CHAPTER 47, we looked at **equivalence relations** and **equivalence structures**. An equivalence relation partitions a base set into cells, and it thoroughly connects up all of the points in a cell. In this chapter, we will continue talking about equivalence structures. But now we will redirect our attention and look inside the cells. We will call the set of points in each cell an **equivalence class**, and we will call the set of all equivalence classes in the structure a **quotient set**.

48.1 Equivalence classes

AN EQUIVALENCE RELATION partitions a base set into cells of equivalent items. For each item in the base set, we can

put together a list (or rather, a set) of all the other items it is equivalent to. We call this set its **equivalence class**. So, the equivalence class of any item x is the set of items that x is equivalent to in the given structure.

We denote the "equivalence class of x" by putting square brackets around x, like this:

$$[x]$$

We can use set builder notation to describe the set too:

$$[x] = \{y \mid x \sim y\}$$

Read that aloud like this: "the equivalence class of x is the set that consists of every y which is such that x is equivalent to y," or more briefly, "the equivalence class of x is the set containing every y that x is equivalent to."

Let us write this down as a definition.

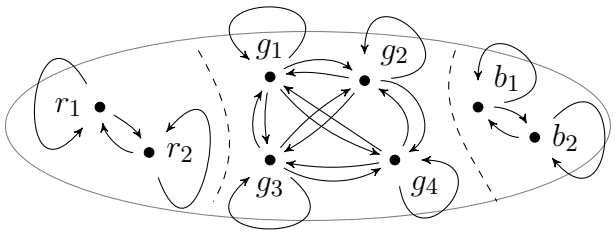
Terminology. The **equivalence class** of an item x is the set of items that x is equivalent to in the given structure. We denote it as $[x]$.

Remark 48.1. Recall from Section 6.5 that **set-builder notation** lets us specify a recipe for building a set. It has the shape: $\{x \mid \text{ recipe }\}$, which we should read as "the set containing each x which satisfies the recipe." In this case, the set is $\{y \mid x \sim y\}$, which we should read as "each y which is such that x is equivalent to y." In other words, this recipe tells us to find every y that x is equivalent to, and put each such y into the set.

Definition 48.1 (Equivalence class)**.** Given a structure $\mathbf{S} = (A, \sim)$ where A is any base set and $\sim$ is an equivalence relation, we will say that the **equivalence class** of each $x \in A$ is the set $\{y \mid x \sim y\}$, i.e., it is the set containing each y that x is equivalent to. We will denote the equivalence class of x like this: $[x]$.

48.2 Example

RECALL THE STRUCTURE $\mathbf{S} = (A, \sim)$ from Chapter **??**. In this structure, we have two red marbles (r_1 and r_2), four green marbles (g_1, g_2, g_3, and g_4), and two blue marbles (b_1, b_2), which are divided into cells by their color:



To build the equivalence class for any item in this structure, we just put together a list of all of the items it is equivalent to, and we dump that list into a set.

For instance, let's take r_1, and build its equivalence class. It is equivalent to the following items:

$$r_1, r_2$$

We can figure this out just by following the arrows. We can see that r_1 has an arrow pointing to itself, and another arrow pointing to r_2, so our list includes r_1 and r_2. Hence, the equivalence class of r_1 (which we write as "$[r_1]$") is the set containing r_1 and r_2:

$$[r_1] = \{r_1, r_2\}$$

Read that aloud like this: "the equivalence class of r_1 is the set containing r_1 and r_2."

We can follow the same procedure to build the equivalence class of r_2. And we can see by following the arrows coming out of r_2 that its equivalent class is this:

$$[r_2] = \{r_1, r_2\}$$

Notice that the equivalence class of r_2 is also just the set of red marbles, since r_2 is equivalent to each of them (including itself). So r_1 and r_2 are the same set. Hence:

Remark 48.2. In our picture, we have a cell for red marbles, a cell for green marbles, and a cell for blue marbles. In each cell, the items are equivalent to each other in color.

Remark 48.3. Notice that r_1 is equivalent to each of the red marbles, including itself. Hence, the equivalence class of r_1 is just the set of all red marbles. And notice that the same goes for r_2.

$$[r_1] = [r_2]$$

Read that like this: "the equivalence class of r_1 is equal to the equivalence class of r_2," or even " the equivalence class of r_1 is identical to that of r_2."

Remark 48.4. Recall from Section **??** that two **sets are equal** if they have the **same members**. In this case, we cans see that $[r_1]$ and $[r_2]$ have the same members, so they are equal.

We can do the same thing for all of the green marbles. The equivalence class of g_1 is the set of items it is equivalent to, which is g_1, g_2, g_3, and g_4 (in other words, it is equivalent to all of the green marbles, including itself):

$$[g_1] = \{g_1, g_2, g_3, g_4\}$$

We can build the equivalence class for g_2, g_3, and g_4 in the same way:

$$[g_1] = \{g_1, g_2, g_3, g_4\}$$
$$[g_2] = \{g_1, g_2, g_3, g_4\}$$
$$[g_3] = \{g_1, g_2, g_3, g_4\}$$

Remark 48.5. We can construct the equivalence class for each item in an equivalence structure by putting together the list of items it is equivalent to, and putting that list into a set. Moreover, equivalent items will have identical equivalence classes.

Notice that all of these equivalence classes are identical. This is because the green marbles are equivalent to each other, so they are all equivalent to the same set of marbles. Hence:

$$[g_1] = [g_2] = [g_3] = [g_4]$$

We could do the same for each blue marble:

$$[b_1] = \{b_1, b_2\}$$
$$[b_2] = \{b_1, b_2\}$$

Which are also the same set, and hence:

$$[b_1] = [b_2]$$

So, to sum up, we can see from this that for each item x in the structure, we can build its equivalence class $[x]$ by putting together the list of items it is equivalent to, and dumping that into a set. Moreover, we can see that all of the items that are equivalent will have identical equivalence classes.

48.3 Quotient Sets

IF WE COLLECT TOGETHER all the equivalence classes from a structure, and then put them into a set, we call that set a **quotient set** of the original base set A.

Terminology. A **quotient set** of a structure $\mathbf{S} = (A, \sim)$ is the set of all equivalence classes of $\mathbf{S}$.

Recall our marbles example. The red marbles belong to one equivalence class (namely, $\{r_1, r_2\}$, the green marbles belong to another equivalence class (namely, $\{g_1, g_2, g_3, g_4\}$, and the blue marbles belong to a third equivalence class (namely, $\{b_1, b_2\}$. So we have three equivalence classes in our structure:

$$\{r_1, r_2\}$$
$$\{g_1, g_2, g_3, g_4\}$$
$$\{b_1, b_2\}$$

Remark 48.6. There are three equivalence classes here, but each one has more than one name. The first one can go by name $[r_1]$ or $[r_2]$, the second one can go by the name $[g_1]$, $[g_2]$, $[g_3]$, or $[g_4]$, and the third one can go by the name $[b_1]$ or $[b_2]$.

If we put these into a set, we get this:

$$\{\{r_1, r_2\}, \{g_1, g_2, g_3, g_4\}, \{b_1, b_2\}\}$$

We call this set the **quotient set** of A under $\sim$. The word "quotient" in casual English conveys some idea of division, or dividing up, and you can use that to remember the idea here. Basically, we are saying that the equivalence relation $\sim$ divides the base set A up into equivalence classes. The result of dividing it up is the set of equivalence classes we have here, so that's why we call it the "quotient" set.

Remark 48.7. In casual English, the word "quotient" usually refers to the result of dividing something up (e.g., dividing one number by another). That's roughly what an equivalence relation does to a set: it divides it up into equivalence classes.

To denote the quotient set, we write this:

$$A/\sim$$

Notation 48.1. To denote the **quotient set** of a structure $(A, \sim)$, we write this: $A/\sim$.

Read that aloud like this: "the quotient set of A under $\sim$," or if you like, you can even read it like this: "the quotient set of A divided up by $\sim$." Hence, we can write out the full quotient set for the marbles example like this:

$$A/\sim \;=\; \{\{r_1, r_2\}, \{g_1, g_2, g_3, g_4\}, \{b_1, b_2\}\}$$

Let's put down a definition for a quotient set.

Definition 48.2 (Quotient set)**.** Given a structure $\mathbf{S} = (A, \sim)$ where A is any base set and $\sim$ is an equivalence relation, we will say that the **quotient set** of A under $\sim$ is the set of all equivalence classes of $\mathbf{S}$. We will denote the quotient set of $(A, \sim)$ like this: $A/\sim$.

As a matter of notation, it can be a little cumbersome to write out all of the equivalence classes in full, when we list out the contents of a quotient set. But recall that we can refer to an equivalence class by putting the name of one of its element in square brackets. For instance, to refer to the equivalence class of r_1, we can write this:

$$[r_1]$$

Notation 48.2. The equilavence class $\{r_1, r_2\}$ has two names: $[r_1]$ and $[r_2]$. Either one of those names can be used when we want to refer to the set $\{r_1, r_2\}$.

We could also write $[r_2]$, since that is another name for the same equivalence class:

$$[r_1] = \{r_1, r_2\} \qquad\qquad [r_2] = \{r_1, r_2\}$$

So, instead of denoting the quotient set like this:

$$A/\sim \;=\; \{\{r_1, r_2\}, \{g_1, g_2, g_3, g_4\}, \{b_1, b_2\}\}$$

We can use any of the bracketed names to refer to the equivalence classes. For instance, we could write it like this:

$$A/\sim \;=\; \{[r_2], [g_3], [b_1]\}$$

We can pick any of the bracketed names we like when we write out a quotient set. For instance, we could also write this, to refer to the same quotient set:

$$A/\sim \;=\; \{[r_1], [g_4], [b_2]\}$$

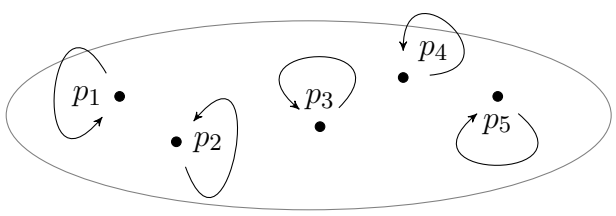
However, we usually pick one of the names as a **representative**, and then we always just use that name. In our case, we can just pick $[r_1]$, $[g_1]$, and $[b_1]$ as our representatives. Then we can say that the quotient set is this:

$$A/\sim \;=\; \{[r_1], [g_1], [b_1]\}$$

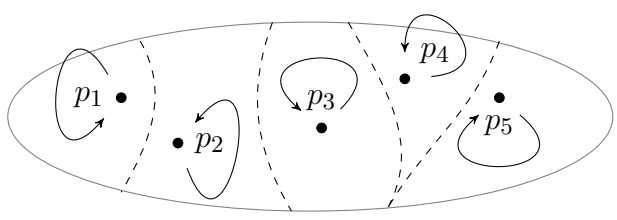
Terminology. Given a set of equal equivalence class names $[x] = [y] = [z] = \ldots$, we can pick any one of them and designate it as the **representative** for this set.

48.4 More Examples

Example 48.1. Let's consider a limit case: where every item in a base set belongs in its own, separate equivalence class. E.g.:



We can see that this partitions the set into cells that contain one item each:



Remark 48.8. Notice what this relation looks like: each item is connected to itself. This is basically just an **identity** relation, i.e., it says each item is identical to itself.

Remark 48.9. If we remember that this relation is just an identity relation, then these equivalence classes make sense. For p_1, which points are identical to it? Just p_1. Which points are identical to p_2? Only p_2. And so on.

So we have a base set, let's call it A, with five points in it:

$$A = \{p_1, p_2, p_3, p_4, p_5\}$$

And we have a relation that connects each item with itself:

$$\sim \; = \; \{(p_1, p_1), (p_2, p_2), (p_3, p_3), (p_4, p_4), (p_5, p_5)\}$$

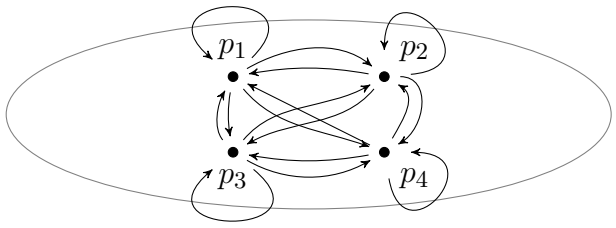
Thus, we have five separate equivalence classes, each of which contains only one item:

$$[p_1] = \{p_1\} \quad [p_2] = \{p_2\} \quad [p_3] = \{p_3\} \quad [p_4] = \{p_4\} \quad [p_5] = \{p_5\}$$

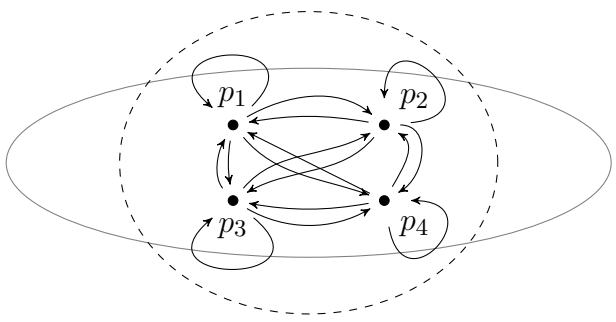
The quotient set for A under $\sim$ is therefore this:

$$A/\sim \; = \; \{[p_1], [p_2], [p_3], [p_4], [p_5]\}$$

Example 48.2. Let's consider the opposite extreme: where every item in a base set belongs in one big equivalence class. For example, consider this structure:



We can see that this partitions the set into a single cell, which contains all of the points:



Remark 48.10. Here we have one cell that contains all of the points, and those points are thoroughly connected. This is a limit case: a 1-cell partition.

So we have a base set A with four points in it:

$$A = \{p_1, p_2, p_3, p_4, \}$$

And we have a relation that identifies all the items:

$$\begin{aligned} \sim \; = \; \{&(p_1, p_1), (p_1, p_2), (p_1, p_3), (p_1, p_4), \\ &(p_2, p_1), (p_2, p_2), (p_2, p_3), (p_2, p_4), \\ &(p_3, p_1), (p_3, p_2), (p_3, p_3), (p_3, p_4), \\ &(p_4, p_1), (p_4, p_2), (p_4, p_3), (p_4, p_4)\} \end{aligned}$$

Remark 48.11. Notice that when we have a cell whose points are **thoroughly connected**, the equivalence relation includes the **product** of the points in that cell. This is because the product is the set of all possible pairings of the points, and to be thoroughly connected is exactly that: to make every possible connection between the points.

The equivalence classes for each point are these:

$$[p_1] = \{p_1, p_2, p_3, p_4\}$$
$$[p_2] = \{p_2, p_2, p_3, p_4\}$$
$$[p_3] = \{p_3, p_2, p_3, p_4\}$$
$$[p_4] = \{p_4, p_2, p_3, p_4\}$$

Remark 48.12. The notation $A/\sim \;= \{[p_1]\}$ makes it obvious what the picture of this structure looks like. Because the quotient set of A under $\sim$ has just one equivalence class, we know that the picture will depict all of the points thoroughly connected together in one cell.

Hence, all of these equivalence classes are identical:

$$[p_1] = [p_2] = [p_3] = [p_4]$$

We can pick any one of these as the representative, so let's just pick $[p_1]$. The quotient set for A under this relation $\sim$ is thus:

$$A/\sim \;= \{[p_1]\}$$

48.5 Summary

IN THIS CHAPTER, we learned about **equivalence classes** and **quotient sets**.

- Given a structure $\mathbf{S} = (A, \sim)$, the **equivalence class** for an element x in A is the set of all elements in A that x is equivalent too (in symbols: it is $\{y \mid y \sim x\}$). We denote the equivalence class of x like this: $[x]$.
- The equivalence class of equivalent elements are identical. E.g., if $x \sim y$, then $[x] = [y]$. For any set of identical equivalence classes, we can pick any one of them as a **representative** for the set.
- A **quotient set** is the set of all equivalence classes for A under $\sim$. We denote it like this: $A/\sim$.

[49]

FURTHER READING

To pursue the topic of equivalence classes and partitions further, the following list may offer some helpful starting points.

- Stewart and Tall (2015, ch. 4) provide an introductory-level discussion of equivalence relations.
- Steinhart (2018, ch. 2) also provides an excellent introduction for the beginner to the topic of equivalence relations.
- Cummings (2020, ch. 9) presents a slow discussion of equivalence relations and how to construct proofs about them.
- Warner (2019, ch. 3) provides a thorough and helpful discussion of equivalence relations, with fully worked examples and proofs.

- Velleman (2019, section 4.5) provides a discussion of equivalence relations and proofs about them.
- Pinter (2014, ch. 3) offers a simple discussion of some of the basic theorems and proofs about equivalence classes and quotient sets.
- Jongsma (2019, ch. 6) provides a rigorous introduction to some of the basic theorems and proofs about equivalence classes.

Part [10]

ORDERED SETS

One of the things we do with a collection of objects is that we put the objects into some sort of order. We arrange them so that this one comes before that one, and that other one comes before this other one, and so on. There are many ways to order objects too. We might only order some of the objects, or we might order all of them. We might put them into a line, or we order them more like a lattice going up the side of a house.

In Part 10, we discuss **ordered sets**, which are sets that are equipped with some kind of additional internal ordering or arrangement. We look at how these structures are defined, we look at how to tell when two such ordered structures are essentially the same, and we look at how to draw pictures of them that represents their ordering structures.

[50]

ORDERINGS

IN THIS CHAPTER, we will begin to look at another class of relational structure: *ordered sets*. An **ordered set** is a structure comprised of two components: (i) a base set, and (ii) an ordering relation.

An **ordering relation** is a relation that puts items in the base set into some kind of order, so that some of the items "come before" others (in the arrangement). An ordering relation is another way we can give structure to sets.

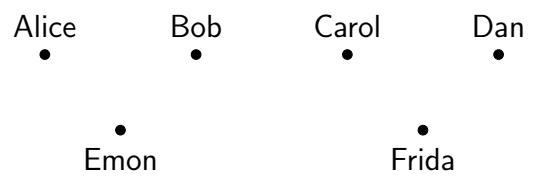
Ponder. What kinds of ordered sets do you encounter in your daily life? What collections of things is it more convenient to treat as ordered rather than unordered?

50.1 Ordered Sets

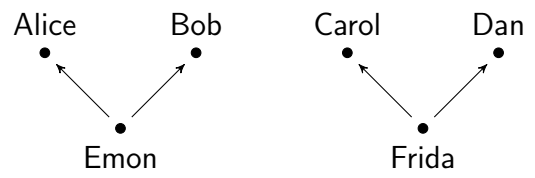
AT A HIGH LEVEL, the concept of **order** amounts to the idea that some items "come before" other items. We put some of

the items "behind" others. For instance, we might arrange the items from "lesser" to "greater," or "earlier" to "later," or whatever other way we might want to order the items.

Let's consider an example. Suppose we have 5 people. Let's call this set of people A, and let's draw a picture of it by representing each person as a point:



Now let's impose some order onto this set. Let's say that Emon is the son of Alice and Bob, while Frida is the daughter of Carol and Dan. To represent this, let's draw some arrows:



Each arrow in the picture represents "being a descendant of." Hence, since we have an arrow from Emon to Alice, that means that Emon is a descendent of Alice. Likewise, since we have an arrow from Emon to Bob, that means Emon is also a descendent of Bob (which of course makes sense, since every child is a descendent of *two* parents).

If we take the set of these arrows all together, we have an ordering **relation** that we can write down as a set of pairs. To see this, let's call this ordering relation R, and let's start writing out the pairs.

First, to encode the arrow that goes from Emon to Alice, let's write down "$(\mathsf{Emon}, \mathsf{Alice})$" to represent that Emon is a descendent of Alice:

Remark 50.1. There are many ways to order things. Smaller to bigger, earlier to later, slower to faster, and so on. At a high level, we can think of all of these different varieties in the same way: they are all instances of arranging some of the items so that they come before others.

Remark 50.2. Recall from Chapter 19 that a **relation** is just a set of ordered pairs. An ordering relation is precisely that: a set of pairs that tell us which items come before which other items.

$$R = \{(\mathsf{Emon}, \mathsf{Alice})\}$$

Notice that the order matters: the first name is the descendent of the second. If we wrote "$(\mathsf{Alice}, \mathsf{Emon})$," we would be saying that Alice is a descendent of Emon, which is not what we see in the picture. What we see in the picture is that Emon is a descendent of Alice, which is why we drew the arrow from Emon to Alice, and why we write "$(\mathsf{Emon}, \mathsf{Alice})$."

Next, to encode the arrow from Emon to Bob, let's also add "$(\mathsf{Emon}, \mathsf{Bob})$" to our list:

$$R = \{(\mathsf{Emon}, \mathsf{Alice}), (\mathsf{Emon}, \mathsf{Bob})\}$$

Finally, let's do the same for Frida and her parents:

$$R = \{(\mathsf{Emon}, \mathsf{Alice}), (\mathsf{Emon}, \mathsf{Bob}) \\ (\mathsf{Frida}, \mathsf{Carol}), (\mathsf{Frida}, \mathsf{Dan})\}$$

Now we have written down exactly the relation we drew in the picture.

However, instead of calling this relation R, a mathematician would prefer to use the "less than" symbol — i.e., "$<$" as when we write "$3 < 4$" (3 is less than 4). This is because the "less than" symbol can be used to denote not just the ordering of numbers, but any ordering whatever. Hence, instead of writing R, we can write "$<$":

$$< \ = \ \{(\mathsf{Emon}, \mathsf{Alice}), (\mathsf{Emon}, \mathsf{Bob}) \\ (\mathsf{Frida}, \mathsf{Carol}), (\mathsf{Frida}, \mathsf{Dan})\}$$

Moreover, to say that Emon is a descendant of Alice, we can just write this:

Notation 50.1. In general, when we encode an ordering relation as a set of ordered pairs with the shape (x, y), this means that x **comes before** y in the ordering arrangement.

Notation 50.2. Actually, mathematicians use a whole family of **similar looking symbols** to refer to ordering relations, depending on the context. For instance, the following are fairly common: $<$ and $\leqslant$, $\prec$ and $\preccurlyeq$, $\sqsubset$ and $\sqsubseteq$, and even $\subset$ and $\subseteq$. We will stick to $<$, and later we will also use $\leqslant$.

$$\text{Emon} < \text{Alice}$$

We can read that aloud like so: "Emon is a descendant of Alice," or even "Emon precedes Alice in this ordering." Similarly, to say that Frida is a descendent of Carol and Frida is a descendent of Bob, we can write this:

$$\text{Frida} < \text{Carol} \qquad\qquad \text{Frida} < \text{Bob}$$

Also, notice that we have a set called "A," and we have a relation "$<$" which arranges the items in A into who is a descendent of who. So, we have a **structure** here. If we were calling the relation R, we would describe our structure like this:

$$\mathbf{S} = (A, R)$$

But since we're referring to the relation as "$<$" we can instead denote the structure like this:

$$\mathbf{S} = (A, <)$$

This structure is called an *ordered set*. An **ordered set** is any structure that consists of a base set and an ordering relation.

Remark 50.3. When we write Emon $<$ Alice, we can see why the "less than" symbol (i.e., "$<$") is appropriate. It tells us that Emon comes *before* Alice in the ordering at hand (namely, ordering by descendents), just like how "$3 < 4$" tell us that 3 comes *before* 4 in the ordering at hand (namely, ordering the numbers into bigger and smaller numbers).

Terminology. An **ordered set** is a structure $\mathbf{S} = (A, <)$ composed of (i) a base set A and (ii) an ordering relation $<$ that arranges items from A by putting them into an order.

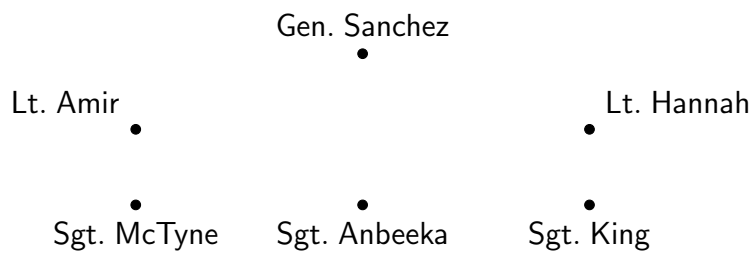
50.2 Partial Orders

AN ORDERING RELATION places the items from a base set into an ordering arrangement. In doing so, it will order *some* of the items. We call such an ordering a **partial order**, because it may not order all of the items in the set. It might arrange only some of them. If it puts all of the elements into the ordering, we say it is a **total order**.

Terminology. A **partially ordered set** is an ordered set *some* of whose elements are ordered. Partially ordered sets are also called **posets**. A **totally ordered set** is an ordered set *all* of whose elements are ordered.

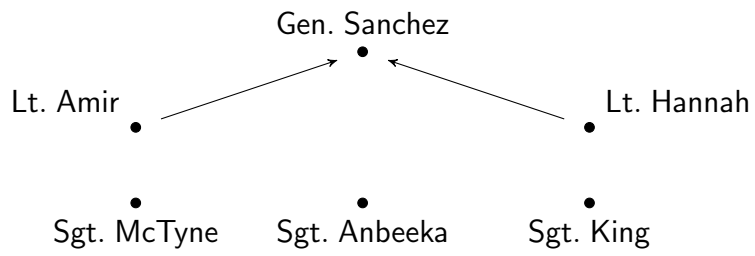
Mathematicians have a short-name for a partially ordered set: they call it a **poset** (pronounced "POE-set" or "PAH-set," depending on preference).

As an example of a poset, suppose that A is a small group from a military unit:



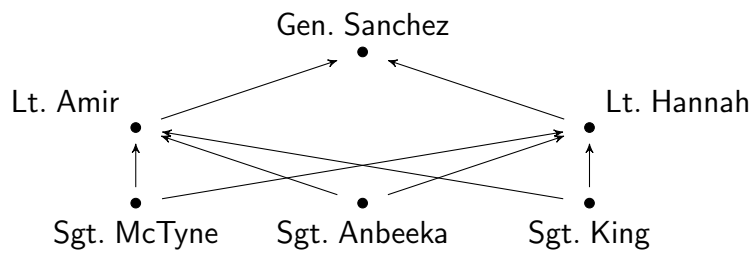
Remark 50.4. The abbreviations "Gen.," "Lt.," and "Sgt." stand for the following ranks: "General," "Lieutenant," and "Sergeant."

Let us impose a partial ordering "$<$" on this set, which represents the chain of command. Let's say that Lieutenants Amir and Hannah each report to General Sanchez:

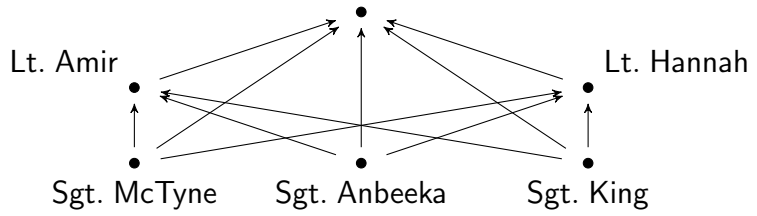


Remark 50.5. A chain of command is a perfectly good example of an ordering relation: some people take orders from others, so some are "below" others in the chain.

Let's also say that Sergeants McTyne, Anbeeka, and King each report to Lieutenants Amir and Hannah:



And of course, let's also say that Sergeants McTyne, Anbeeka, and King each report to General Sanchez as well:



Now we have drawn in all the arrows that represent the chain of command. Each arrow represents an instance of this "reports to" or "takes orders from" relation.

For example, Lieutenant Amir reports to (takes orders from) General Sanchez, and Sergeant King reports to (takes orders from) Lieutenant Hannah. Hence, we can write:

$$\text{Lt. Amir} < \text{Gen. Sanchez} \qquad \text{Sgt. King} < \text{Lt. Hannah}$$

Read the one on the left like so: "Lieutenant Amir reports to General Sanchez," or even "Lieutenant Amir is below General Sanchez in this chain of command." Read the one on the right in a similar way.

Let's write down the full "chain of command" relation as a set of pairs. Since Sergeant McTyne reports to Lieutenant Amir, let's write down that pair first:

$$< \;=\; \{(\text{Sgt. McTyne}, \text{Lt. Amir})\}$$

Since Sergeant McTyne reports to Lieutenant Hannah and to General Sanchez as well, let's add those pairings too:

$$\begin{aligned} < \;=\; \{&(\text{Sgt. McTyne}, \text{Lt. Amir}), (\text{Sgt. McTyne}, \text{Lt. Hannah}), \\ &(\text{Sgt. McTyne}, \text{Gen. Sanchez})\} \end{aligned}$$

Remark 50.6. Notice that some of the individuals in this picture report to more than one person. For instance, Sergeant McTyne reports to both Lieutenant Amir and Lieutenant Hannah. But notice also that some people don't have any relation to each other in this ordering. For instance, Sergeant McTyne and Sergeant Anbeeka have no relation: neither one reports to the other or takes orders from the other. So, some of the items in this ordered set are **incomparable**: they cannot be compared under the "$<$" relation.

Remark 50.7. When we write "Lt. Amir $<$ Gen. Sanchez," it is clear again why the "less than" symbol (namely "$<$") is appropriate. It tells as that Lieutenant Amir comes *before* (or is *lower* on the chain than) General Sanchez.

If we continue like this and write down all the pairings in our picture, we get this:

$$\begin{aligned} < \; = \; \{&(\text{Lt. Amir}, \text{Gen. Sanchez}), (\text{Lt. Hannah}, \text{Gen. Sanchez}),\\ &(\text{Sgt. McTyne}, \text{Lt. Amir}), (\text{Sgt. McTyne}, \text{Lt. Hannah}),\\ &(\text{Sgt. McTyne}, \text{Gen. Sanchez}), (\text{Sgt. Anbeeka}, \text{Lt. Amir}),\\ &(\text{Sgt. Anbeeka}, \text{Lt. Hannah}), (\text{Sgt. Anbeek;}, \text{Gen. Sanchez})\\ &(\text{Sgt. King}, \text{Lt. Amir}), (\text{Sgt. King}, \text{Lt. Hannah}),\\ &(\text{Sgt. King}, \text{Gen. Sanchez})\} \end{aligned}$$

Together, A and $<$ make up a structure, i.e., an ordered set:

$$\mathbf{S} = (A, <)$$

This is a **partially ordered set** (a poset), because some of the people in the base set don't take orders from some of the others. For instance, neither Lieutenants Amir and Hannah take orders from the other (there is no arrow between them), and likewise, none of the Sergeants take orders from each other either.

Remark 50.8. A **poset** puts only some of the elements from the base set into the ordering arrangement. It may leave some of the items without any ordering between them at all. Hence, in a partially ordered set, some of the elements in the base set may be **incomparable**.

50.3 Total Orders

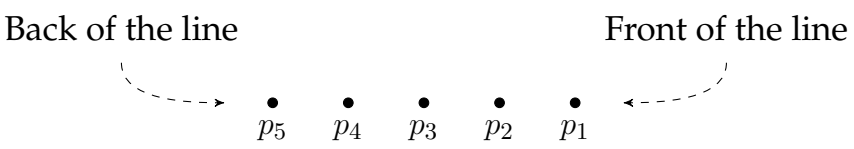
AS AN EXAMPLE of a **totally ordered set**, imagine 5 people standing in line at the ice cream truck. The person at the front of the line is able to order ice cream, and all the others behind them are waiting their turn.

There is an ordering to this line, which goes as follows. The first person in line does not have to wait for anybody to buy ice cream, the second person in line gets to go after the first person gets their ice cream, the third person in line gets

Remark 50.9. In a **totally ordered set**, every element is placed in the ordering arrangement. What this means is that every two elements are ordered with respect to each other: either the one comes before the other in the arrangement, or the other comes before the latter. There simply are no two elements in a totally ordered set that are **incomparable**.

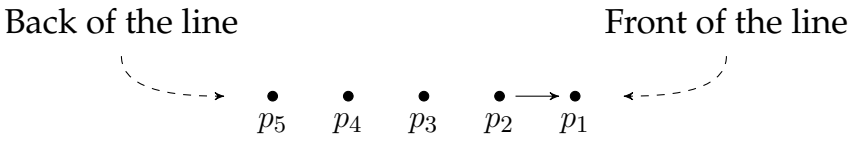
to go after both the first and the second get their ice cream, and so on.

Let A be the set of people, and let $<$ be the ordering. We can draw this with a picture. First, let's draw the base set:

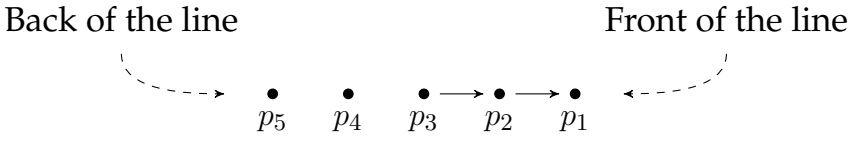


Remark 50.10. Strictly speaking, we don't need to indicate which side is the back of the line and which side is the front of the line, since it will be obvious once we put in all the ordering arrows. We're just including these markers as hints.

To represent who has to wait for who, let's draw an arrow. For instance, p_1 is at the front of the line, and p_2 has to wait for them to get their ice cream, so let's draw an arrow from p_2 to p_1 to represent that p_2 comes behind p_1 in this ordering:

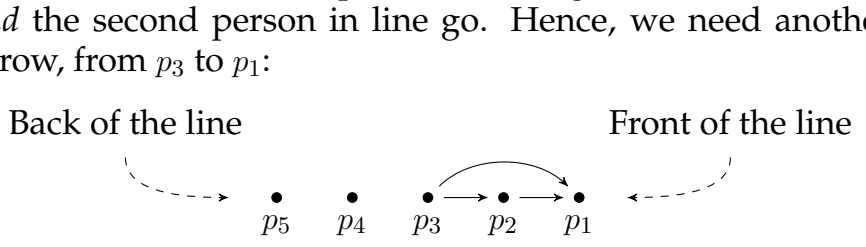


Next, p_3 has to wait for p_2 to get their ice cream, so let's put an arrow in to represent that too:

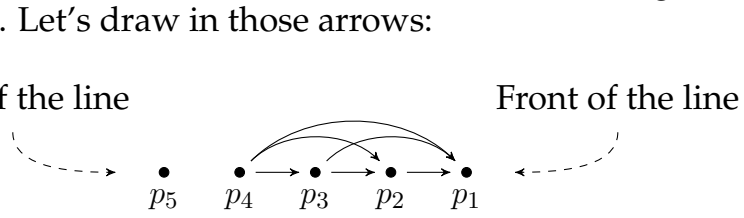


Remark 50.11. Notice that this ordering is **transitive**: if x precedes y in the line and y precedes z in the line, then x precedes z too.

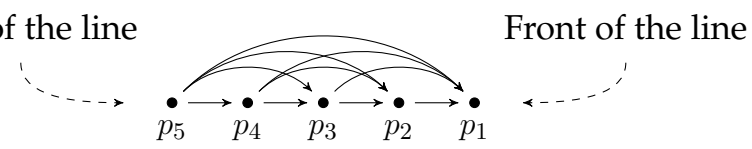
However, p_3 also has to wait for p_1 to get their ice cream too. That's the idea behind waiting in line. Each person gets their turn. So the third person cannot go until *both* the first *and* the second person in line go. Hence, we need another arrow, from p_3 to p_1:



In a similar fashion, p_4 has to wait for everybody in front of them, so p_4 has to wait for each of p_3, p_2, and p_1 to get their ice cream. Let's draw in those arrows:



Finally, p_5 has to wait for each of the people in front of them to get their ice cream too, so:



With that, we have drawn in all the arrows that indicate who waits for who. We have an ordered set:

$$\mathbf{S} = (A, <)$$

This is a **total ordering**, because each person is related to every other person in this ordering. That's why there are so many arrows. Each person has to wait for every other person in front of them! Hence, p_3 stands behind p_2, but also p_1:

$$p_3 < p_2 \qquad\qquad p_3 < p_1$$

And p_4 stands behind all those in front of them:

$$p_4 < p_3 \qquad p_4 < p_2 \qquad p_4 < p_1$$

Even p_1 is related to every other person in the line, since p_1 stands in front of each of the others.

Remark 50.12. Notice that every two distinct points are related by this ordering. Pick any two distinct points, and you will see that there is an arrow between them. For instance, pick p_4 and p_2. There is an arrow between them (it goes from p_4 to p_2). The same goes for, say, p_5 and p_3, or p_5 and p_2, or any other two distinct points in the picture.

Terminology. A **total ordering** is an ordering that orders *all* elements in the base set. When we say it "orders all elements," we mean that there is an arrow between every two points. None of the points are incomparable (i.e., none of them are missing an arrow between them).

50.4 Summary

IN THIS CHAPTER, we learned about **ordered sets**.

- An **ordered set** is a structure $\mathbf{S} = (A, <)$ where A is a set and $<$ is a relation that places items from A into an ordering arrangement. To say that x precedes y in the ordering, we can write this: $x < y$.
- We call an ordered set **partially ordered** if it orders *some* of the elements (i.e., for at least some x and y from the base set, either $x < y$ or $y < x$). A partially ordered set need not order every element in the base set though. There may be an x and a y that are not **comparable** (i.e., neither $x < y$ nor $y < x$). Partially ordered sets are also called **posets**.
- We call an ordered set **totally ordered** if it orders *all* of the elements (i.e., if for every x and y in the base set, either $x < y$ or $y < x$).

[51]

STRICT VS. NON-STRICT

IN CHAPTER 50, we introduced **ordered sets**. An ordered set is a structure comprised of a base set and an ordering relation. The ordering relation puts the items from the base set into some kind of ordering.

There are two distinct kinds of orderings: one is **strict**, and the other is **non-strict**. We will discuss both of these in this chapter, since they have different properties.

51.1 Two Kinds of Ordering

THERE ARE TWO KINDS of orderings, which roughly correspond to the symbols "$<$" and "$\leqslant$" that we use in arithmetic. In arithmetic, we can write down expressions with this shape:

Ponder. Strict ordering corresponds to, say, "$3 < 4$," as in "3 is strictly less than 4," whereas non-strict ordering corresponds to "$3 \leqslant 4$," as in "3 is less than or equal to 4." Can you generalize these two kinds of comparisons to examples that don't involve numbers? E.g., what would "$<$" and "$\leqslant$" mean if used with, say, people in the chain of command?

$$x \leqslant y$$

We can read that like this: "x is less than or equal to y." Another way to read it is this: "x is *no bigger* than y." The expression asserts that x is either the same number as y, or it is some smaller number, but it is not a bigger number than y.

In arithmetic, we can also write down expressions that have this shape:

$$x < y$$

We can read that like this: "x is less than y." Another way to read that is this: "x is *strictly* less than y." The expression asserts that x is definitely smaller than y (x cannot be the same as y, it must be smaller).

Corresponding to these two symbols are two kinds of orderings, called **strict** and **non-strict** orderings. We will use the symbol "$<$" to denote a **strict** ordering (where x is strictly prior to y in the ordering arrangement), and we use the symbol "$\leqslant$" to denote a **non-strict** ordering (where x is at least as prior in the arrangement as y is, but may also be equal to y in the arrangement).

Strict and non-strict orderings have different properties. In the next couple of sections, we'll look at the properties of each of these types of orders.

Example 51.1. "$3 \leqslant 4$" is true, because 3 is no bigger than 4. "$4 \leqslant 4$" is also true, because the number on the left (namely, 4) is no bigger than the number on the right (in fact, they are equal). "$3 < 4$" is true, because 3 is smaller than 4. But "$4 < 4$" is false, because 4 is not smaller than 4.

Terminology. "$x \leqslant y$" lets x be anything up to and *including* y, whereas "$x < y$" lets x be anything up to but *not including* y. A **strict ordering** ("$<$") puts items from the base set into a strict prior-posterior arrangement. A **non-strict ordering** ("$\leqslant$") puts the items into a prior-or-equal-to arrangement.

51.2 Strict Order

A STRICT ORDERING requires that the ordering relation put the items from the base set into a strict prior-and-posterior arrangement. What are the properties of such an arrangement?

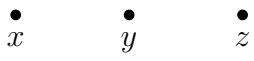
A strict ordering must be **transitive**, **asymmetric**, and **irreflexive**. Let's take a little bit of time to look at each of these in turn, to see why these properties are right.

51.2.1 Transitivity

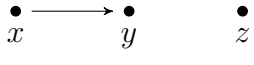
An ordering relation needs to be **transitive**. Why is this? The basic concept of an ordering seems to require this. Ordering involves some notion of chaining. If x is behind y in the ordering, and y is behind z in the ordering, then x is behind z as well.

Remark 51.1. Recall from Chapter 20 that a transitive relation is one that connects up the endpoints of every two-arrow chain. That is, if x is connected to y and y is connected to z, then x is connected to z too.

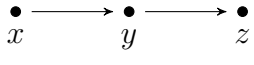
To see this, take any three points in a base set:



Now suppose that in the ordering, x is prior to y in the arrangement:



Suppose also that y is prior to z in the arrangement too:



Remark 51.2. Think about our intuitive understanding of an ordering. Think of, say, people standing in line. If any three people in line (call them x, y, and z), if $x < y$ and $y < z$, what does your intuitive understanding of ordering tell you about x and z? How is x related to z?

So, in this ordering, we have these two things:

$$x < y \qquad\qquad y < z$$

Our intuitive understanding of ordering suggests that if indeed this is true, then x would be prior to z in the ordering arrangement too:

$$x < z$$

This really does match with our intuitive understanding of ordering. Think about it. If three people are standing in

line, the third person in line does not have to wait only for the second person. They have to wait for the first person in line too.

Hence, we should draw in a further arrow here, to match this idea we have about ordering:

This is why we say that a strict ordering must be *transitive*. Items at the back of the line stand behind not just the next item in line. They stand behind every other item going up the line.

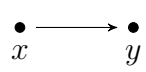
51.2.2 Asymmetry

A strict ordering must be **asymmetric**. Why is this? Well, if you think about it, the whole concept of an ordering is antithetical to symmetry. A symmetrical relationship is a two-way connection: both points are related to each other in the same way, like a mirror image.

Remark 51.3. Recall from Chapter 21 that an **asymmetric** relation is a relation where for any x and y in the base set, if x is connected to y, then y is not connected back to x. There simply are no two-way arrows at all.

But ordering is the opposite of that. To put two items into an order is precisely to arrange them so that one stands behind the other!

Think about this visually. Suppose we have two points x and y, and we have arranged it so that x comes before y in the ordering:



It would make no sense to *also* say that y comes before x in the ordering. I.e., it would make no sense for there to be a two way arrow such as this:

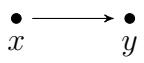
If we had a pair of two-way arrows like this, we'd have to say that both x and y are "prior" to the other, and that makes no sense. An ordering is one-way, by definition, so its got to be one or the other.

In a strict ordering, we should see only one-way arrows between two elements, to indicate the direction of the order.

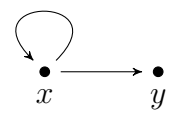
51.2.3 Irreflexivity

A strict ordering must be **irreflexive**. Why is this? Because the ordering is *strict*. If we say $x < y$, we mean that x is *strictly* prior to y in the ordering arrangement. We are not saying $x \leqslant y$, where x is allowed to be prior, or *equal to* y. No, we are saying that x must definitely be prior to y in the ordering arrangement.

To see this, suppose we have two points where the first is prior to the second in the ordering arrangement:



In this picture, it is clear that x is *strictly prior* to y, because there is just one arrow from x to y, and so we can see that x comes before y in this ordering arrangement. But what would happen if there were an arrow from x back to itself? Like this:



Well, remember that each arrow means "is strictly prior to" (in the ordering arrangement). Hence, the arrow from x to y means "x is strictly prior to y," or "$x < y$."

But this also applies to the arrow from x to x. An arrow from x to x would mean "x is strictly prior to x," or "$x < x$."

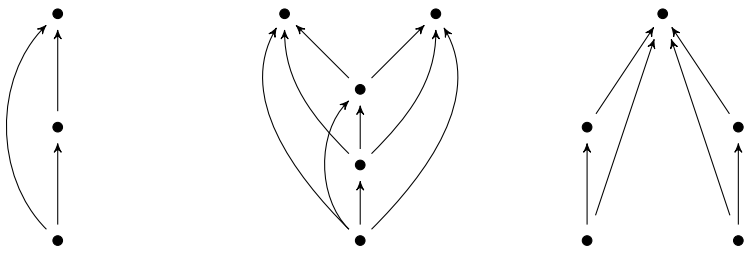
Remark 51.4. Recall from Chapter 21 that an **irreflexive** relation is a relation where no x is related to itself. There simply are no points with arrows that loop back to themselves.

But that makes no sense when we are talking about a *strict* order. Nothing can be strictly prior to itself.

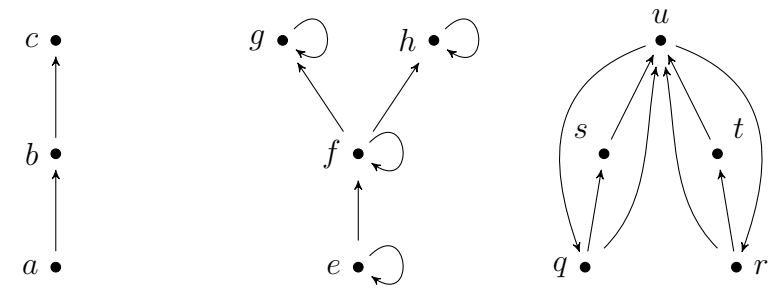
This makes it clear why there will be no self-loops in a strict ordering. A strict ordering arrow can only go from a point to some other point. It can never go back to itself.

51.2.4 Summing Up

To sum up, if we look at pictures of strict orderings, the *only* arrows we should see are one-way arrows, arranged into transitive chains, with no self-loops. Here are three examples:



Here are examples of relations that are *not* strict orderings:



Example (i) Example (ii) Example (iii)

Example (i) is not a strict ordering because it is not transitive. We can see that a is behind b in the line (i.e., $a < b$), and we can see that b is behind c in the line (i.e., $b < c$), but a is not behind c (i.e., $a \not< c$)! This does not accurately model our intuitive understanding of ordering. If Tom were behind Sally

Remark 51.5. It is a good idea to manually check each of these examples of strict orderings, and confirm for yourself that each is a strict ordering. To do that, check that each one is (a) irreflexive (i.e., make sure there are no self-loops), (b) asymmetric (i.e., make sure there are no two-way arrows), and (c) transitive (i.e., make sure there is an arrow from the start- to the end-point of every two-arrow chain). Likewise, it is a good idea to manually check the examples that fail to be strict orderings, and identify exactly where they fail to exhibit the properties of a strict ordering. Which properties do they fail to have?

and Sally were behind Gina in line, then surely Tom would be behind Gina in line too. So, whatever sort of relation Example (i) is, it is not a strict ordering.

Example (ii) is not a strict ordering for two reasons. First, it is not transitive (for instance, $e < f$ and $f < g$ but $e \not< g$). Second, there are self-loops (i.e., it is reflexive). For example, there is an arrow from f to f. If this were a strict ordering, that would mean that in this picture, f would stand strictly behind f in the ordering (i.e., $f < f$). But that does not fit with how we normally understand a strict ordering. Nothing can be strictly behind itself in line. Hence, whatever sort of relation Example (ii) is, it is not a strict ordering.

Example (iii) is not a strict ordering either because there are some two-way arrows in it. For example, there is an arrow from q to u, and another arrow from u back to q. If this were a strict ordering, that would mean that q is strictly behind u in the ordering, and also that u is strictly behind q in the ordering. But that makes no sense. Ordering goes one way. So whatever sort of relation Example (iii) is, it is not a strict ordering either.

51.3 Non-strict Order

IN A STRICT ORDERING, $x < y$ means that x is strictly prior to y in the ordering arrangement. With a *non-strict* ordering, $x \leqslant y$ means that x is no farther forward than y is in the ordering arrangement. To put it another way, it means that x is *at least* as far back in the arrangement as y is.

Ponder. When we shift our attention from a strict ordering to a non-strict ordering, what properties of the ordering do we need to change, to accommodate the non-strict version?

How do we characterize a non-strict ordering relation? A non-strict ordering relation is **reflexive**, **antisymmetric**, and **transitive**. Let's look at each of these in turn, to see why these properties are the ones that characterize non-strict ordering.

51.3.1 Transitive

A non-strict ordering must be **transitive**, for the same reasons that a strict ordering are. We are trying to model our intuitive understanding of ordering here, and as we normally understand it, ordering is transitive. So, to match the way we understand ordering, we insist that ordering relations are transitive, and this applies to both strict and non-strict relations.

Remark 51.6. As we normally understand ordering, if Tom is behind Sally in line, and Sally is behind Gina in line, then Tom is behind Gina too. We understand ordering as a transitive relation (and this applies to both **strict** and **non-strict** orderings).

51.3.2 Reflexive

A non-strict ordering must be **reflexive**. Contrast this with a strict ordering. As we noted earlier, a strict ordering must be *irreflexive*, because nothing can be strictly prior to itself.

But with a *non-strict* ordering, $x \leqslant y$ doesn't mean that x is *strictly* prior to y. It only means that x is at least as far back in the ordering as y. Hence, x can be equal to y in the ordering.

But then, we can say $x \leqslant x$, because it's true that x is at least as far back in the ordering arrangement as itself. And this will be true for any x from the base set. No matter which x we pick in the base set, it will be true that $x \leqslant x$, because each one will be at least as far back in the ordering as itself.

Hence, every point in the base set needs to have an arrow looping back to itself. In contrast to strict orderings (which have *no* self-loops), in a *non-strict* ordering, *every point* must have a self-loop.

Remark 51.7. Recall from Chapter 21 that a relation is **reflexive** if it relates every item x in the base set back to itself.

Remark 51.8. In strict orderings, no points will have a self-loop, because it is never true that $x < x$. In a non-strict ordering, every point will have a self-loop, because it is always true that $x \leqslant x$.

51.3.3 Antisymmetric

A non-strict ordering must be **antisymmetric**. This follows from its reflexivity. Since each point has a self-loop arrow, it is possible that $x \leqslant x$, and $x \geqslant x$.

Note that this is not possible with a strict ordering. With a strict ordering, it is not possible for $x < x$ nor $x > x$, because there are no self-loops. But here, with a non-strict ordering,

Remark 51.9. Recall from Chapter 21 that a relation is **antisymmetric** if the only way that any given x and y can be symmetrically related to each other is if they are the same point. So, if $x \leqslant y$ and $y \leqslant x$, then $x = y$.

because each point has an arrow that loops back to itself, this is possible.

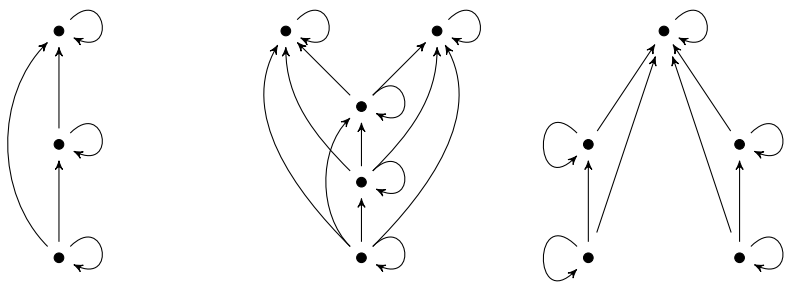
However, it's only possible for one and the same point. As we normally understand an ordering, it is one-directional, and not two-way. So $x \leqslant y$ and $y \leqslant x$ should never occur in a (non-strict) ordering, except for the cases where x and y are the same point.

But that is the very definition of antisymmetry: namely, to be such that the only way for $x \leqslant y$ and $y \leqslant x$ is if $x = y$. And that is what we have here, with a non-strict ordering. So a non-strict ordering relation is antisymmetric.

Remark 51.10. Ordering goes one way, and because of this both strict and non-strict orderings are **not symmetric**. However, they are not symmetric in different ways. In a **non-strict** ordering, the only way that $x \leqslant y$ and $y \leqslant x$ can both be true is when $x = y$. Hence, a non-strict ordering is **anti**symmetric. In a **strict** ordering, it is never the case that $x < y$ and $y < x$. Hence, a strict ordering is **a**symmetric.

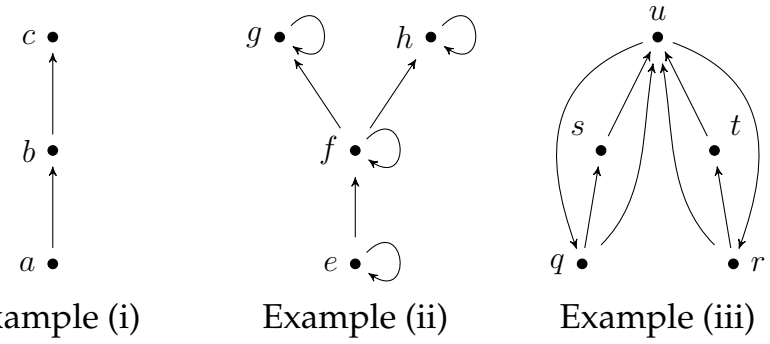
51.3.4 Summary

To sum up, if we look at pictures of non-strict ordering relations, we should see a self-loop on every point, no two-way arrows, and transitive chains. Here are three examples of non-strict orderings:



Remark 51.11. As before, it is a good idea to manually check each of these examples of non-strict orderings, and confirm for yourself that each is a non-strict ordering. To do that, check that each one is (a) reflexive, (b) antisymmetric, and (c) transitive.

Here are some examples that are *not* non-strict orderings:



Example (i) Example (ii) Example (iii)

Remark 51.12. Likewise, it is a good idea to manually check the examples that fail to be non-strict orderings, and identify exactly where they fail to exhibit the required properties. Which properties do they fail to have?

Example (i) fails to be a non-strict ordering because it is not reflexive, and it is not transitive. If it were a non-strict ordering, we would see a self-loop on each point, and we would see an arrow from a to c.

Example (ii) fails to be a non-strict ordering because it is not transitive. If it were a non-strict ordering, we would see an arrow from e to g, and also from e to h.

Example (iii) fails to be a non-strict ordering because it is not reflexive, and it is not antisymmetric. If it were a non-strict ordering, we would see a self-loop on each point, and we wouldn't have two way arrows (such as the ones from q to u and u back to q, or r to u and u back to r).

Terminology. A **strict ordering relation** is any relation that is irreflexive, asymmetric, and transitive. A **non-strict ordering relation** is any relation that is reflexive, antisymmetric, and transitive. We will denote strict ordering relations with the "$<$" symbol, and we will denote non-strict ordering relations with the "$\leqslant$ symbol.

51.4 Definitions

NOW THAT WE HAVE EXPLORED the properties that characterize both the strict and the non-strict versions of ordering relations, let us write down some definitions. First, let's put down a definition for strict and non-strict ordering relations.

Definition 51.1 (Ordering relations)**.** Given a base set A and a relation R on A, we will say that R is a **strict ordering relation** if it is irreflexive, asymmetric, and transitive. We will denote strict ordering relations with the symbol "$<$," and we will write "$x < y$" to denote that x is strictly prior to y in the ordering arrangement.

We will say that R is a **non-strict ordering relation** if it is reflexive, antisymmetric, and transitive. We will denote non-strict ordering relations with the symbol "$\leqslant$," and we will write "$x \leqslant y$" to denote that x is at least as prior in the ordering arrangement as y.

Most mathematicians use the word "order" or "ordering" to refer to the **non-strict** version of ordering. We will too. Hence, if we speak of an "ordering relation," unless we say otherwise, we will mean "$\leqslant$." If we want to speak about a strict ordering relation, we will be explicit and say a "*strict* ordering relation," or we will just use the symbol "$<$."

Terminology. By convention, most mathematicians are thinking of the **non-strict** version when they speak merely about "order" and don't specify whether they mean the strict or non-strict version. We will follow suit.

Next, let us define strictly and non-strictly ordered sets. These are base sets equipped with either a strict ordering relation, or a non-strict ordering relation.

Definition 51.2 (Ordered sets)**.** Given a base set A and a strict ordering relation $<$, we will say that a **strictly ordered set** is a structure $\mathbf{S} = (A, <)$ comprised of the base set A and the ordering relation $<$.

Given a base set A and a non-strict ordering relation $\leqslant$, we will say that a **non-strictly ordered set** is a structure $\mathbf{S} = (A, \leqslant)$ comprised of the base set A and the ordering relation $\leqslant$.

Terminology. We will refer to a structure $\mathbf{S} = (A, <)$ as a **strictly ordered set**, and we will refer to a structure $\mathbf{S} = (A, \leqslant)$ as a **non-strictly ordered set**.

If we speak of an "ordered set," unless we say otherwise, we will mean a non-strictly ordered set, i.e., $\mathbf{S} = (A, \leqslant)$. If we want to speak about a strictly ordered set, we will be explicit and say a "*strictly* ordered set," or we will be explicit and use the symbols: $\mathbf{S} = (A, <)$.

Terminology. By convention, when we speak merely about an **ordered set** and do not specify whether we mean the strict or non-strict version, we will mean the **non-strict** kind.

Let us finally turn to putting down a definition for partially and ordered sets. First, let us say that two elements x and y in an ordered set are *comparable* (or *incomparable*) in the ordering if they are (or are not) related by the ordering, i.e., if either $x \leqslant y$ or $y \leqslant x$ (or neither $x \leqslant y$ nor $y \leqslant x$). Likewise for a strict ordering with $<$.

Terminology. In an ordered set, two items x and y are **comparable** if either $x \leqslant y$ or $y \leqslant x$. If neither $x \leqslant y$ nor $y \leqslant x$, then x and y are **incomparable**. Likewise for $<$: they are **comparable** if $x < y$ or $y < x$, whereas they are **incomparable** if neither $y < y$ nor $y < x$.

Definition 51.3 (Incomparability)**.** Given an ordered set $\mathbf{S} = (A, \leqslant)$ and any two elements $x, y \in A$, we will say that x and y are **comparable** if either $x \leqslant y$ or $y \leqslant x$, and we will say that x and y are **incomparable** if neither $x \leqslant y$ nor $y \leqslant x$. Similarly, any two elements x and y from a strictly ordered set $\mathbf{S} = (A, <)$ are comparable or incomparable in the same way, but using $<$ rather than $\leqslant$.

Now we can put down a definition for partially and totally ordered sets:

Terminology. By default, every ordered set is a **partially ordered set** (or a **poset** for short). If every pair of elements x and y from the set are **comparable**, then we say it is a **totally ordered set**. In a partially ordered set, it may be the case that there are elements x and y that are incomparable. This will never be true in a totally ordered set. Every two items are comparable in a totally ordered set.

Definition 51.4 (Partially and totally ordered sets)**.** Given an ordered set $\mathbf{S} = (A, \leqslant)$ or $\mathbf{S} = (A, <)$, we will say that $\mathbf{S}$ is a **partially ordered set** (or a **poset**) by default. We will say that $\mathbf{S}$ is **totally ordered** if every x and y from A are comparable.

51.5 Summary

IN THIS CHAPTER, we learned about the strict and non-strict versions of orderings.

- A **strict ordering** is any relation that is irreflexive, asymmetric, and transitive. A **non-strict ordering** is any relation that is reflexive, antisymmetric, and transitive.
- A **strictly ordered set** is a structure $\mathbf{S} = (A, <)$ comprised of a base set A and a strict ordering relation $<$. A **non-strictly ordered set** is a structure $\mathbf{S} = (A, \leqslant)$ comprised of a base set A and a non-strict ordering relation $\leqslant$.
- By convention, when we speak of "order" or "ordering," we will mean the non-strict kind. If we want to speak about a strict ordering, we will explicitly say so and use the appropriate $<$ symbol.
- A **partially ordered set** (be it strict or non-strict) is one that puts at least some of the items from the base set into an ordering arrangement. A **totally ordered set** (be it strict or non-strict) is one that puts every item from the base set into the ordering arrangement.

[52]

HASSE DIAGRAMS

IN THIS CHAPTER, we will look at a special way to draw ordered sets. It is a lot more compact than the graphs we have been drawing. These diagrams of ordered sets are called **Hasse diagrams**.

52.1 The Diagrams

THERE IS A SPECIAL KIND OF DIAGRAM that we can use to draw ordered sets. It is called a **Hasse diagram** (pronounced "HASS-uh diagram").

In a Hasse diagram, we draw chains of dots and lines going upwards. The upwards direction shows the ordering, and the chaining shows which dots are chained.

Terminology. A **Hasse** diagram is a special kind of diagram that we use to draw ordered sets. In a Hasse diagram, we draw the dots in order from bottom to top, and we connect the dots with lines to show chains.

Because we draw these diagrams with the ordering going upwards, we can always tell if two dots x and y are ordered, just by looking at the diagram. The rule is this: if one dot x is lower than another dot y in a chain, then $y \leqslant x$.

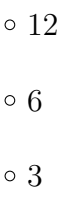
52.2 Examples

Example 52.1. Suppose we have an ordered set $\mathbf{S} = (A, \leqslant)$, where $A = \{3, 6, 12\}$, and $\leqslant$ totally orders these numbers from smallest to biggest. Hence:

$$6 \leqslant 3 \qquad 12 \leqslant 6 \qquad 12 \leqslant 3$$

To draw a Hasse diagram, we first draw the dots in order going upwards:

Remark 52.1. We always draw the order going **upwards**, so we put 3 at the bottom, and 12 at the top of the picture.



Then we draw lines to show the chaining:

Remark 52.2. We connect up all items in a **chain** to show transitivity. So, we draw a line between 3 and 6, and then between 6 and 12. This also connects up 3 and 12, by putting them in the same chain.

We can see that we have a single chain here, going from 3 up through 12. We also know that any dot lower in the chain is prior to any dot higher in the chain. Hence, we can see just

from the picture that $6 \leqslant_3$, and $12 \leqslant_6$. Also, we can tell that $12 \leqslant_3$, because 3 is lower than 12 in the same chain.

This Hasse diagram is much simpler than the graphs we have been drawing. The graph drawing has lots more arrows. Look at the two side by side:

Graph Hasse diagram

The Hasse diagram is a lot simpler, and it contains the same information. Basically, when we draw a Hasse diagram, we only draw the chains, and we let our readers use their imagination to fill in the self-loops and the other transitive arrows.

Remark 52.3. With a Hasse diagram of a non-strict order $\leqslant$, we let the reader fill in the details about reflexivity (each point has a self-loop) and transitivity (lower items are connected to higher items in the same chain).

Example 52.2. Let's look at another example. Suppose we have an ordered set $\mathbf{S} = (A, \leqslant)$, where the base set is this:

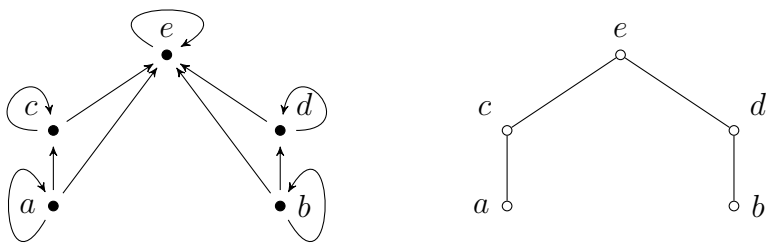
$$A = \{a, b, c, d, e\}$$

And the (non-strict) ordering is this:

$$\begin{aligned} \leqslant \; = \; \{&(a,a),(b,b),(c,c),(d,d),(e,e),\\ &(a,c),(a,e),(b,d),(b,e),(c,e),(d,e)\} \end{aligned}$$

Remark 52.4. It is a good idea to try and draw (i) a graph, and (ii) a Hasse diagram of this ordered set on your own. Then compare your drawings with the pictures below.

Here is the graph and the Hasse diagram of this ordered set, side by side:



Graph Hasse diagram

Remark 52.5. Recall from Section 51.4 that two points x and y in an ordered set are **incomparable** if neither $y \leqslant_x$ nor $x \leqslant_y$. In this diagram, what are all the pairs of incomparable points?

Again, it is obvious that the Hasse diagram is simpler, and yet it tells us the same information. For example, we can see that $c \leqslant_a$ and $e \leqslant_c$, but we can also see that $e \leqslant_a$ because a and e are in the same chain, and a is lower than e in the chain. We can also see that, for instance, a and b are **incomparable**.

Example 52.3. Let $\mathbf{S} = (A, \leqslant)$ be an ordered set where the base set A is defined like this:

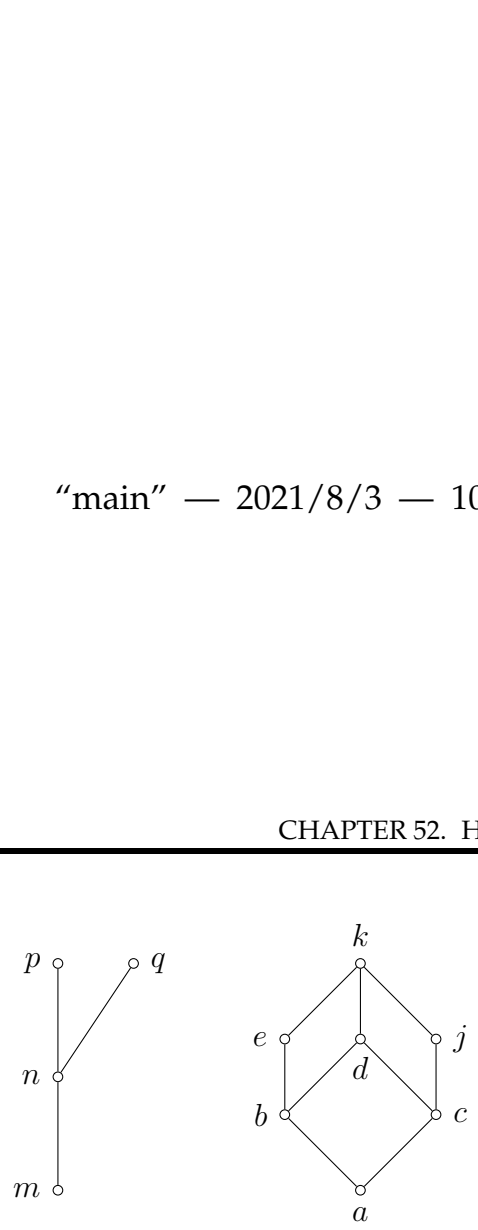
$$A = \{a, b, c, d, e, j, k, m, n, p, q\}$$

Remark 52.6. Again, it would be a good idea to try and draw a graph and a Hasse diagram of this ordered set on your own, first. Then compare your drawings with the pictures below.

And the ordering relation $\leqslant$ is defined like this:

$$\begin{aligned}\leqslant \;=\; \{&(a,a),(b,b),(c,c),(d,d),(e,e),(j,j),\\ &(k,k),(m,m),(n,n),(p,p),(q,q),\\ &(a,b),(a,d),(a,e),(a,k),(a,c),(a,j),\\ &(b,d),(b,e),(b,k),(d,k),(e,k),\\ &(c,d),(c,j),(c,k),(j,k),\\ &(m,n),(m,p),(m,q),(n,p),(n,q)\}\end{aligned}$$

The Hasse diagram for this ordered set looks like this:



We can see from the picture that this ordered set has quite an odd structure. It actually has two disjoint pieces. None of the items in the left piece are comparable with any in the right piece. We might think of this as an ordering structure that has two parallel ordering tracks.

Remark 52.7. It is obvious that all of the items in the left piece are **incomparable** with all of the items in the right piece. But there are incomparable points inside each piece too. Can you find all the dots that are incomparable in the left piece? How about the right piece?

52.3 Summary

IN THIS CHAPTER, we looked at **Hasse diagrams**, which are special ways to draw ordered sets.

- In a Hasse diagram, we draw the items in the base set in order, going upwards, and we connect the dots with lines to show chains.
- The basic rule is this. Suppose x and y are dots in the same chain (i.e., if there is an unbroken chain of dots and lines between them). If x is lower on the chain than y, then $x \leqslant y$ in the ordering.
- We do not draw reflexive self-loops, and transitive arrows. Instead, we let the reader fill these in with their own imagination.

[53]

ORDER ISOMORPHISMS

IN THIS CHAPTER, we will look at how to tell if two ordered sets have the same shape — i.e., are **isomorphic**. Two ordered sets are isomorphic if there is an **order isomorphism** between them, i.e., a one-to-one mapping of the points from the first set to the points of the second set, which **preserves the order**.

Ponder. Can you come up with a precise definition of what it means for two ordered sets to have the **same shape** or **structure**?

53.1 Maps between Ordered Sets

WE CAN CONSTRUCT a **map** (a **function**) from one set to another set, by mapping each point from the first set to some point in the second set. Of course, we can do the same thing with ordered sets, by mapping the points in their base sets.

Let's do an example. Suppose we have two ordered sets:

$$\mathbf{S} = (A, \leqslant) \qquad \mathbf{T} = (B, \leqslant)$$

Well, the ordering relation in each of these structures might be different. So maybe we shouldn't use the same symbol $\leqslant$ for both. For the sake of clarity, let's add a subscript to each "$\leqslant$" symbol, so that we can tell them apart. To the ordering for A, let's add "A" as a subscript:

$$\leqslant_A$$

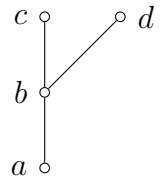
That way, when we use this symbol, it's obvious that the ordering we are referring to is the one that belongs to the set A. We can do the same thing for the ordering that belongs to B, by writing it like this: $\leqslant_B$.

So, let's say instead that we have two ordered sets, like this:

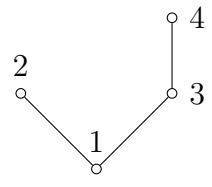
$$\mathbf{S} = (A, \leqslant_A) \qquad \mathbf{T} = (B, \leqslant_B)$$

Notation 53.1. By adding a subscript to each ordering relation symbol, it makes it clear that "$\leqslant_A$" denotes the ordering on the set A that we have in $\mathbf{S}$, while "$\leqslant_B$" denotes the ordering on the set B that we have in $\mathbf{T}$.

Now suppose that these ordered sets looks like this:



$\mathbf{S} = (A, \leqslant_A)$



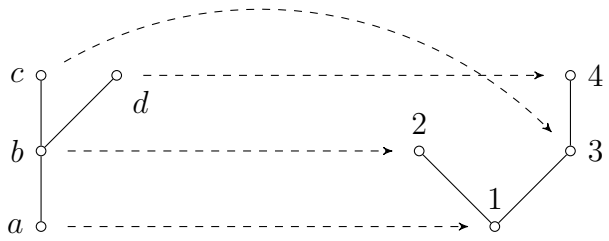
$\mathbf{T} = (B, \leqslant_B)$

Let's build a map (function) from $\mathbf{S}$ to $\mathbf{T}$. Let's call our function $f : \mathbf{S} \to \mathbf{T}$, because it is a map from the ordered set (structure) $\mathbf{S}$ to the ordered set (structure) $\mathbf{T}$.

To build such a map, all we have to do is build function between the **base sets** of these two structures. And we can

pick any mapping between the base sets that we like. For instance, let's map the points from A to B like this:

$$f : \mathbf{S} \to \mathbf{T}$$



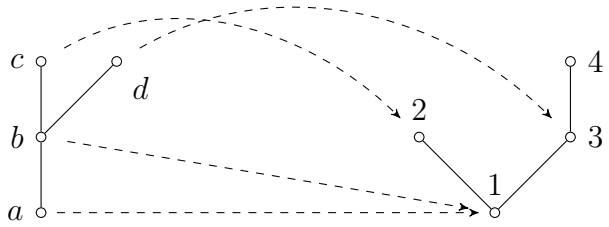
$\mathbf{S} = (A, \leqslant_A)$ $\qquad$ $\mathbf{T} = (B, \leqslant_B)$

That is to say, let's define f like this:

$$f(a) = 1 \qquad f(b) = 2 \qquad f(c) = 3 \qquad f(d) = 4$$

We can come up with a different mapping, if needed. For example, we might construct a new map g, like this:

$$g : \mathbf{S} \to \mathbf{T}$$



$\mathbf{S} = (A, \leqslant_A)$ $\qquad$ $\mathbf{T} = (B, \leqslant_B)$

That is, we might define g like this:

$$g(a) = 1 \qquad g(b) = 1 \qquad g(c) = 2 \qquad g(d) = 3$$

Remark 53.1. Notice that f maps every point from $\mathbf{S}$ to a distinct point in $\mathbf{T}$. Hence, f is **injective**. Also, f covers every point in $\mathbf{T}$ (no points in $\mathbf{T}$ are left without an arrow pointing to them). Hence, f is **surjective** too.

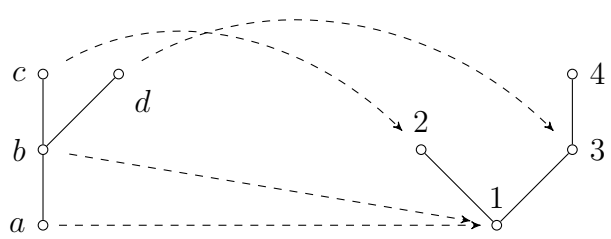
Remark 53.2. Notice that g maps a and b to the same point 1. Hence, g is not **injective**. Also, g does not cover all of the points in $\mathbf{T}$ (in particular, nothing is mapped to 4). Hence, g is not **surjective**.

53.2 Order-preserving Maps

Terminology. Given two ordered sets $\mathbf{S}$ and $\mathbf{T}$, a map from $\mathbf{S}$ to $\mathbf{T}$ is an **order-preserving map** if it keeps the ordering intact when we follow the arrows from $\mathbf{S}$ to $\mathbf{T}$. If x comes before y in the ordering of $\mathbf{S}$, then the point that x is mapped to in $\mathbf{T}$ should come before the point that y is mapped to in $\mathbf{T}$ as well.

NOT ALL MAPS BETWEEN STRUCTURES are equal. Some preserve the ordering. A map from one ordered set to another **preserves the order** of the first set if it keeps the ordering intact when we use the map and follow the points from the one set to the other. If the points are ordered in the first set, then the points at the other end of the mapping should have the same ordering.

Consider the map g from earlier. Here it is, pictured again:



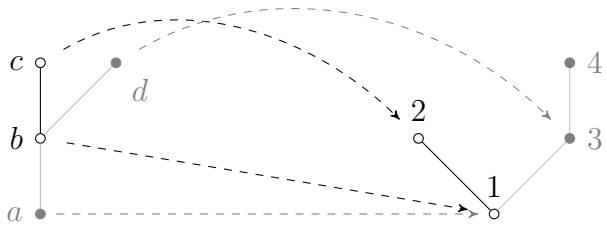
Remark 53.3. To **prove** that a map from one ordered set to another is order-preserving, we need to show that it preserves the order of *every* pair of points in the first set.

Is g an **order-preserving** map? To check, we need to make sure that g preserves the ordering of every pair of points in $\mathbf{S}$.

First, let's check b and c. We can see from the picture that $b \leqslant_A c$ in $\mathbf{S}$. Let's see if g preserves this order. What does g map b and c too? That is, what are $g(b)$ and $g(c)$? If we follow the arrows, we can see that they are these:

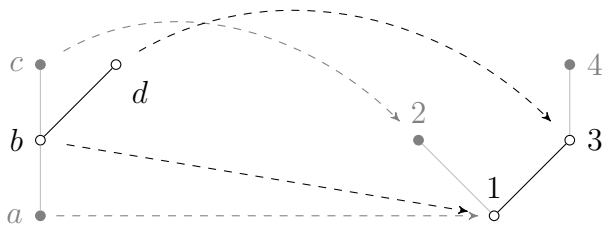
$$g(b) = 1 \qquad g(c) = 2$$

Let's highlight just this part of the mapping in the picture, so we can see it clearly:



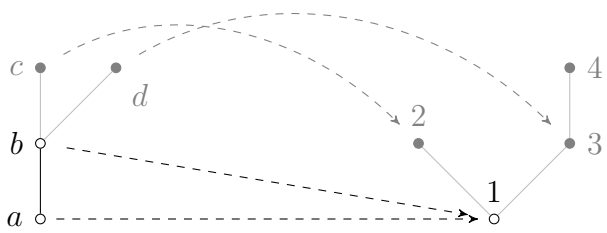
Are the points 1 and 2 in the same order as b and c? Yes they are. In **S**, we can see that $b \leqslant_A c$, and in **T**, we can see that $1 \leqslant_B 2$. So, we can conclude that g does preserve the ordering between b and c when we follow the arrows over to **T**. To put it concisely: $b \leqslant_A c$, and also $g(b) \leqslant_B g(c)$.

Next, let's check b and d. What does g map b and d too? What are $g(b)$ and $g(d)$? Here is the mapping:



We can see that g preserves the order between b and d as well. In **S**, $b \leqslant_A d$, and in **T**, $g(b) \leqslant_B g(d)$, i.e., $1 \leqslant_B 3$.

Next, let's check a and b. Does g preserve the order between a and b? Let's look at the picture:



Remark 53.4. We can see that g maps $b \mapsto 1$ and $c \mapsto 2$. But b is prior to c in the ordering of **S**, and 1 is similarly prior to 2 in the ordering of **T**. So g preserves the ordering of b and c.

Remark 53.5. We can see that g maps $b \mapsto 1$ and $d \mapsto 3$. But b is prior to d in the ordering of **S**, and 1 is similarly prior to 3 in the ordering of **T**. So g preserves the ordering of b and d as well.

Remark 53.6. We can see that g maps $a \mapsto 1$ and $b \mapsto 1$. But a is no higher than b in the ordering of **S**, and 1 is similarly no higher than 1 (itself) in the ordering of **T**. So we can see that g preserves the ordering of a and b too.

We can see that g maps a and b to the same point in $\mathbf{T}$, namely 1. In $\mathbf{S}$, $a \leqslant_A b$, but is $g(a) \leqslant_B g(b)$ in $\mathbf{T}$? Yes, it is, because $f(a) = 1$ and $f(b) = 1$, and $1 \leqslant_B 1$. So f preserves the order between a and b too.

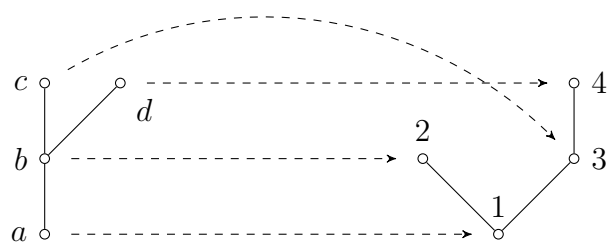
We can see from all this that g does indeed preserve the order of $\mathbf{S}$ in the way that it maps the points to $\mathbf{T}$. The ordering on the left side of the arrows is preserved over on the right side of the arrows, so g is an **order-preserving map**.

Let's put down a definition for order-preserving maps. All we need to say is that if we have two ordered sets $\mathbf{S}$ and $\mathbf{T}$, and we have a map $f : \mathbf{S} \to \mathbf{T}$, then f is an order-preserving map if it preserves the order of every two points.

Definition 53.1 (Order-preserving maps)**.** For any two ordered sets $\mathbf{S} = (A, \leqslant_A)$ and $\mathbf{T} = (B, \leqslant_B)$, and any map $f : \mathbf{S} \to \mathbf{T}$, we will say that f is an **order-preserving map** if whenever $x \leqslant_A y$, then $f(x) \leqslant_B f(y)$ also.

Example 53.1. Let's look at a map that fails to preserve the ordering. Consider this mapping again:

$$f : \mathbf{S} \to \mathbf{T}$$



$\mathbf{S} = (A, \leqslant_A)$ $\mathbf{T} = (B, \leqslant_B)$

To show that a map is not order-preserving, we only need to find one pair of points from $\mathbf{S}$ where it fails to preserve

Remark 53.7. To be completely thorough, we actually need to check that g preserves the order for *every* pairing. So we also need to check whether g preserves the ordering between a and c, and also between a and d. But if you work it out, you will see that it does. In $\mathbf{S}$, $a \leqslant_A c$, and in $\mathbf{T}$, it is also true that $g(a) \leqslant_B g(c)$, i.e., $1 \leqslant_B 2$. Likewise for a and d.

Remark 53.8. To **prove** that a map is not order-preserving, we only need to find **one** pair of points from the first structure whose order is not preserved.

the order. We can do this with b and d. Let's focus on the mapping of b and d:

Does f preserve the ordering? No it does not. We can see in the picture that, in $\mathbf{S}$, $b \leqslant_A d$, yet in $\mathbf{T}$, it is not the case that $f(b) \leqslant_B f(d)$. In fact, $f(b)$ (namely, 2) and $f(d)$ (namely, 4) are **incomparable**. So, f does not preserve the ordering, and hence f is *not* an order-preserving map.

53.3 Order Isomorphisms

AS WE SAW IN CHAPTER 17, two sets A and B are **isomorphic** if they have the same shape, which we denote by writing:

$$A \cong B$$

We know that A and B have the same shape if we can construct an **isomorphism** between them (i.e., a **reversible bijective function**). Such a function shows us that each element from the first set has a counterpart "twin" in the other set (and vice versa). Hence, the two sets differ only in the names of their elements.

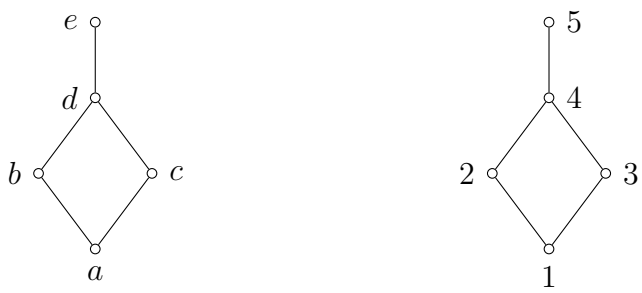
Ordered sets can also be isomorphic (i.e., have the same shape) in this way. An isomorphism between ordered sets is much the same as an isomorphism between unordered sets,

Remark 53.9. Recall from Chapter 16 that a function f is **bijective** if it is both injective and surjective, i.e., it puts the elements from the domain into a one-to-one correspondence with the elements of the codomain. Recall from Chapter 17 that f is **reversible** only if we can construct an **inverse** function f^{-1} that maps the end-points of f right back to their starting points. An **isomporphism** is just a reversible bijective function.

except that it must *also* **preserve the ordering** (and its inverse must preserve the ordering too).

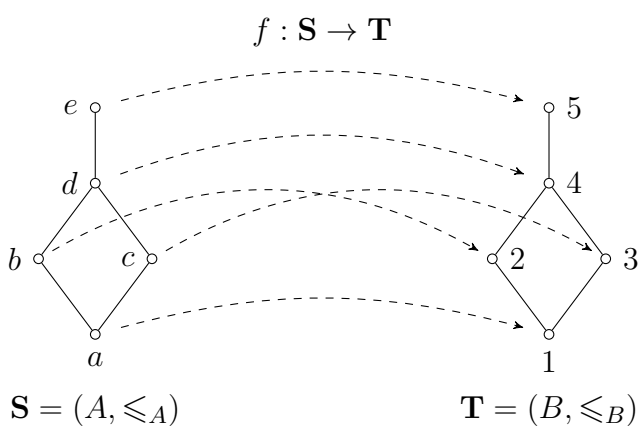
Hence, two ordered sets are isomorphic if we can construct an **order-preserving** isomorphism between them, i.e., a reversible bijective function that preserves their orderings in both directions.

Example 53.2. Consider these two ordered sets:



$\mathbf{S} = (A, \leqslant_A)$ $\quad$ $\mathbf{T} = (B, \leqslant_B)$

Let's add an order-preserving bijective map from $\mathbf{S}$ to $\mathbf{T}$:

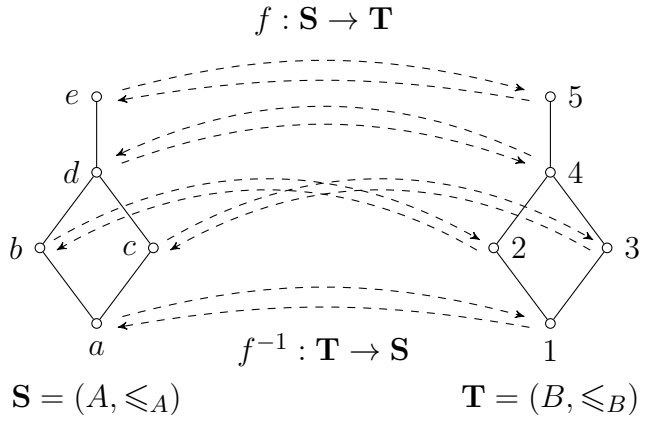


$\mathbf{S} = (A, \leqslant_A)$ $\quad$ $\mathbf{T} = (B, \leqslant_B)$

This map is (i) bijective, and (ii) order preserving. But it is also **reversible**. We can see this because we can construct an order-preserving inverse $f^{-1} : \mathbf{T} \to \mathbf{S}$ which maps all the end-points of f right back to their starting points:

Remark 53.10. We can tell just by looking at these two structures that they have the same shape. But can we be more **precise** about what it means to say that they "have the same shape"? Yes, we can, by using the notion of an order-isomorphism. If each point in the left set has a counterpart "twin" in the right set that stands in exactly the same ordering arrangement, then it is clear that these two ordered sets have exactly the same structure (i.e., they are **isomorphic**).

Remark 53.11. Recall from Chapter **??** that a function f is **reversible** if we can construct an **inverse** function, which maps the endpoints of f back to their starting points.



Hence, f is an **isomorphism**. And since there is an isomorphism between $\mathbf{S}$ and $\mathbf{T}$, we can conclude that $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic**, i.e., $\mathbf{S} \cong \mathbf{T}$.

Remark 53.12. In fact, f is an isomorphism, but so is f^{-1}, because f is the **inverse** of f^{-1}. Hence, we can also say that $\mathbf{T} \cong \mathbf{S}$.

Let's write down a definition for order isomorphisms.

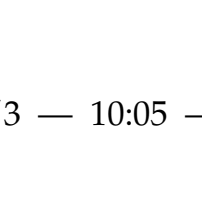
Definition 53.2 (Order isomorphism)**.** For any ordered sets $\mathbf{S}, \mathbf{T}$ and any map $f : \mathbf{S} \to \mathbf{T}$, we will say that f is an **order isomorphism** if f is a reversible, bijective function that preserves the ordering of $\mathbf{S}$, and whose inverse $f^{-1} : \mathbf{T} \to \mathbf{S}$ preserves the ordering of $\mathbf{T}$.

If there is an order isomorphism between two structures $\mathbf{S}$ and $\mathbf{T}$, then we say they are **isomorphic**, or that they are the same **up to isomorphism**. To denote this, we write $\mathbf{S} \cong \mathbf{T}$. Let's put this down as a definition too.

Terminology. Given two ordered sets $\mathbf{S}$ and $\mathbf{T}$, a function $f : \mathbf{S} \to \mathbf{T}$ is an **order isomorphism** if it is a reversible, bijective, order-preserving map. If there is an isomorphism between $\mathbf{S}$ and $\mathbf{T}$, then $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic**, which we denote as $\mathbf{S} \cong \mathbf{T}$.

Definition 53.3 (Isomorphic Ordered Sets)**.** For any ordered sets $\mathbf{S}$ and $\mathbf{T}$, we will say that $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic** (or synonymously: the same **up to isomorphism**) if there is an order isomorphism $f : \mathbf{S} \to \mathbf{T}$. To denote that $\mathbf{S}$ and $\mathbf{T}$ are isomorphic, we will write this: $\mathbf{S} \cong \mathbf{T}$.

Example 53.3. Let us look at an example of an order-preserving map that is *not* an isomorphism. Consider the following two ordered sets:

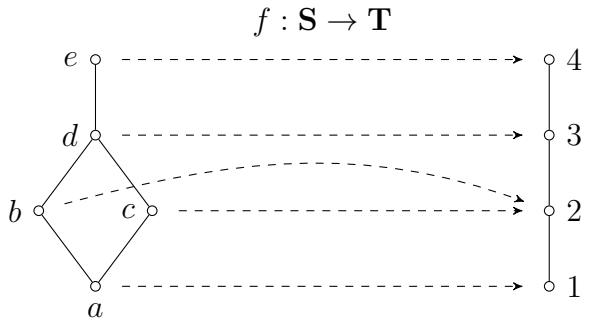


$\mathbf{S} = (A, \leqslant_A)$

$\mathbf{T} = (B, \leqslant_B)$

Remark 53.13. If you squint hard enough at this picture, you can see that $\mathbf{S}$ can be mapped to $\mathbf{T}$ and the order can be preserved. All that we need to do is collapse b and c. Picture putting thumb and forefinger on the outsides of b and c, and then squeezing the diamond shape until it collapses inwards and b and c are smashed together.

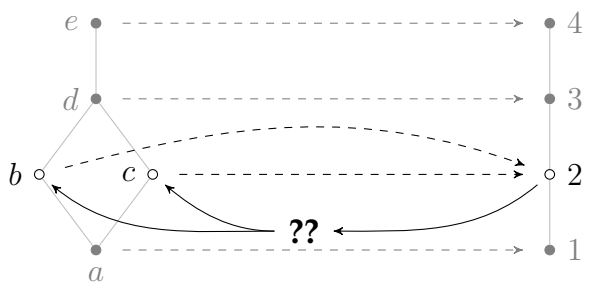
Let's add an order-preserving map from $\mathbf{S}$ to $\mathbf{T}$:



$\mathbf{S} = (A, \leqslant_A)$ $\mathbf{T} = (B, \leqslant_B)$

Remark 53.14. Notice that all of the points from $\mathbf{S}$ are mapped to a distinct "parallel" point in $\mathbf{T}$ except for b and c. Those two points are mapped to one and the same point in $\mathbf{T}$, namely 2. Hence, f is not **bijective**. It therefore cannot be reversed, because it is impossible to draw a single arrow from 2 back to both b and c.

Notice that f is an order-preserving map (verify this for yourself). But it is **not reversible**, because we cannot construct an inverse: f "collapses" b and c into the same point in $\mathbf{T}$ (namely, 2), so it is impossible to reverse 2. There is no way to draw a single arrow from 2 back to both b and c.



Remark 53.15. When we try to construct an inverse function that maps the points from the right side back to the left side, we run into a problem when we have to draw an arrow from 2 back to the left side. Do we send 2 to b, or c?

Hence, f is not an isomorphism. In fact, there cannot be an isomorphism between $\mathbf{S}$ and $\mathbf{T}$ at all. They have a different number of points, so we cannot construct a bijective function between them.

Remark 53.16. Isomorphisms can only be constructed between ordered sets that have the same number of points. If one set has fewer points than the other, then some part of the structure will have to be **collapsed**. This is why isomorphisms must be bijective.

If there is an isomorphism between two ordered sets, then this means that each element in the one set has a counterpart "twin" in the other set, standing in exactly the same ordering relationships. Hence, if two ordered sets are isomorphic, they are essentially the **same structure**. It's just that the elements have different names. Isomorphisms give us a precise way to identify when two ordered sets are structurally identical, without getting sidetracked by names.

53.4 Summary

In this chapter, we learned about **order isomorphisms** and **isomorphic** ordered sets.

- Given two ordered sets $\mathbf{S} = (A, \leqslant_A)$ and $\mathbf{T} = (B, \leqslant_B)$, a map $f : \mathbf{S} \to \mathbf{T}$ is an **order isomorphism** if it is a reversible, bijective, order-preserving map. The inverse $f^{-1} : \mathbf{T} \to \mathbf{S}$ must also preserve the order going the other direction.
- If there is an order isomorphism between two ordered sets $\mathbf{S}$ and $\mathbf{T}$, then $\mathbf{S}$ and $\mathbf{T}$ are **isomorphic**, which we denote like this: $\mathbf{S} \cong \mathbf{T}$.

[54]

FURTHER READING

To pursue ordered sets further, the following list may offer some helpful starting points.

- Velleman (2019, ch. 4) offers an introductory survey of the concept of order and relations, along with a thorough discussion of proofs about them.
- Warner (2019, chs. 3, 5) provide a helpful discussion of ordered sets and how ordering is defined for the different sets of numbers. There are fully worked examples and proofs.
- Jongsma (2019, ch. 7) covers the basic theorems and proofs that underlie the mathematical concept of order.
- Pinter (2014, ch. 4) provides a simple but rigorous discussion of ordered sets. Once you’re comfortable read-

ing simple proofs, this is a good chapter to try.

- Davey and Priestley (1990) provides thorough coverage of many classic topics studied under the mathematical concept of order, along with how these ideas are used in a variety of applications.

Part [11]

COMPUTERS

Computers seem like incredible machines, given all that they can do. But, have you ever heard that it's all just ones and zeros under the hood? How is that possible? How is it possible that mere ones and zeros could give rise to everything we see computers doing today?

In Part 11, we discuss how **computers** actually work, under the hood. We will look at different ways to **represent numbers**, and we will look at a few common number encodings, including **hexadecimal** and **binary** encodings. We will look at the parts and components that are used to build a **CPU**, and we will look at how it is that a CPU actually **executes** a program.

[55]

REPRESENTING NUMBERS

WE HAVE AN IDEA of what the natural numbers are, but if we want to work with them, we need some way to write them down. We need some way to encode them, so that we can store them and transmit them back and forth. In this chapter, we'll look at one particularly efficient way to encode or represent numbers.

55.1 Tally Systems

ONE VERY SIMPLE WAY to write down numbers is to use a tally system, much like we did with the natural numbers. In a tally system, you write down a number by writing down mark after mark after mark on the page.

So, to write, say, three, I put down three marks. To write fourteen, I add fourteen marks. To write two hundred and seventy nine, I put down two hundred and seventy nine marks.

Obviously, this way of writing down numbers is inefficient, especially for big numbers. It takes up a lot of space to write down any numbers that are bigger than, say, ten or twenty.

55.2 Rotating Systems

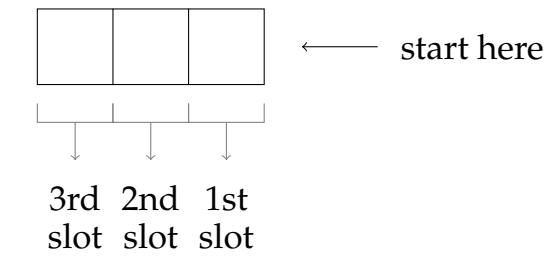
THERE IS ANOTHER WAY to write down numbers, which is much more efficient. Basically, we pick a small set of symbols to serve as representatives of digits, and we cycle through them. When we run out of symbols, we add another column or slot to write new digits in, and we cycle through our symbols again.

Let's build a simple example of this kind of system, so we can see how it works. To start, let's decide on a set of symbols that we want to use to represent digits. Let's say that our set of symbols consists of the following:

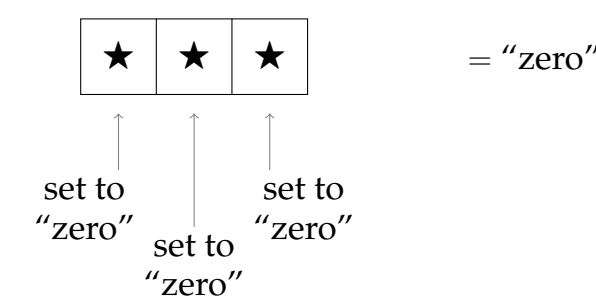
$$\bigstar, \spadesuit, \clubsuit$$

These are in order. So, the lowest digit is $\bigstar$ (we'll think of this as representing no digits at all, or "zero" digits), the next higher digit is $\spadesuit$, and then the highest digit is $\clubsuit$.

Next, let's lay out an array of slots that we can fill with our digit symbols. The first slot goes on the right, and then we add more slots on the left. We'll start with three slots for now, but we can add more later if we need. Here is our array of slots:

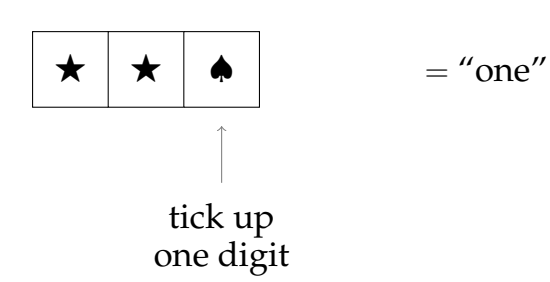


Next, let's fill every slot with our lowest digit symbol (the star), because that digit symbol represents "zero." In other words, let's set all of our slots to "zero":



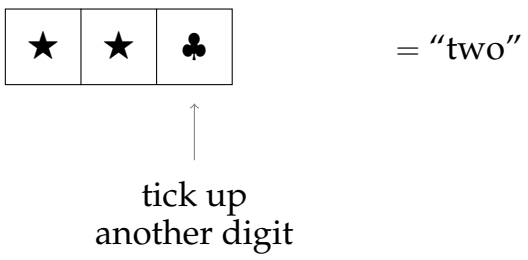
This is how we write the number "zero." We have our array of slots, all of which are set to "zero."

Now that we know how to write zero, let's write the number one. To do that, we increment the digit in the far right slot. So, we tick the rightmost slot up, by changing it from the star symbol to the next digit symbol, which is the spade symbol:



This is how we write the number "one." All of the slots are zero, except for the far right slot, which has ticked its digit up to the spade.

Next, let's write the number two. To do this, we tick up the far slot one more time. So, we change it from the spade, to the next digit symbol, which is the club:

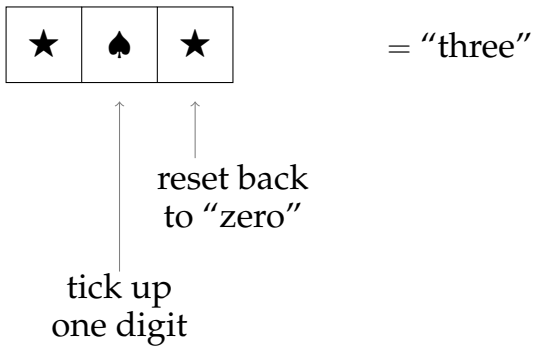


Okay, so we know how to write the first three numbers with our system:

★ ★ ★ = "*zero*"
★ ★ ♠ = "*one*"
★ ★ ♣ = "*two*"

How do we write the next number (which is the number three)? We've run out of digit symbols to use in the first slot. We can't tick up that first slot any higher. So what do we do next?

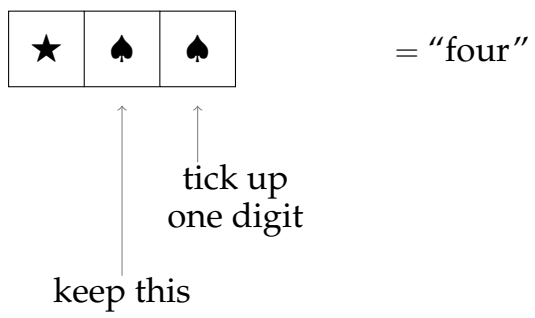
What we do is this: we reset the first slot back to "zero" (the star symbol), and then we increment the 2nd slot by one digit (so we tick it up to the spade):



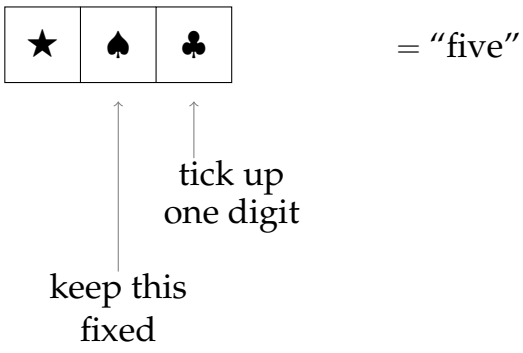
That's how we write the number "three." We can't write "three" using only the 1st slot, because we've run out of digit symbols. So we have to increment the 2nd slot, and then we can start over in the first slot.

This is what we do whenever we run out of digits in a slot. We just reset that slot back to "zero," and then we increment the slot to the left by one digit symbol. Once we have done that, we can start incrementing our initial slot again.

Next, let's do the number "four." To do this, we keep the second slot fixed, and we tick up the first slot:



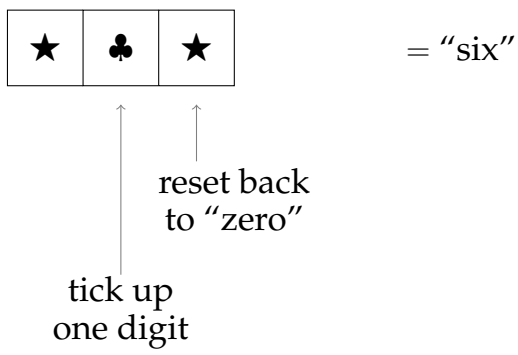
To do "five," we again keep the second slot fixed, and we increment the first slot one more time:



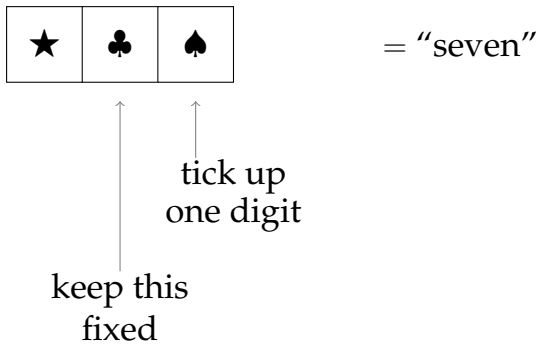
Now we know how to write the first six numbers with our system:

★ ★ ★	= "*zero*"	★ ♠ ★	= "*three*"
★ ★ ♠	= "*one*"	★ ♠ ♠	= "*four*"
★ ★ ♣	= "*two*"	★ ♠ ♣	= "*five*"

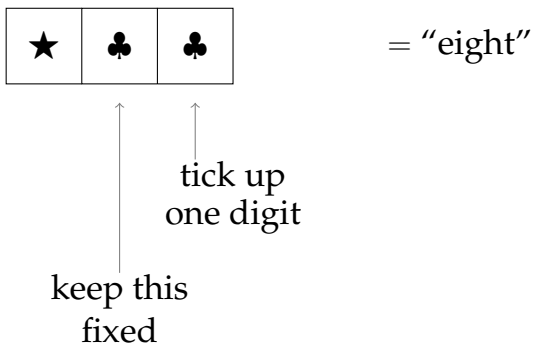
How do we write the next number (which is the number six)? We've again run out of digit symbols to use in our first slot, so what do we do? At this point, we do much the same thing that we did the last time that we ran out of digit symbols. We ticked up the slot to the left. So, we reset the first slot to "zero" and we tick up the second slot one digit:



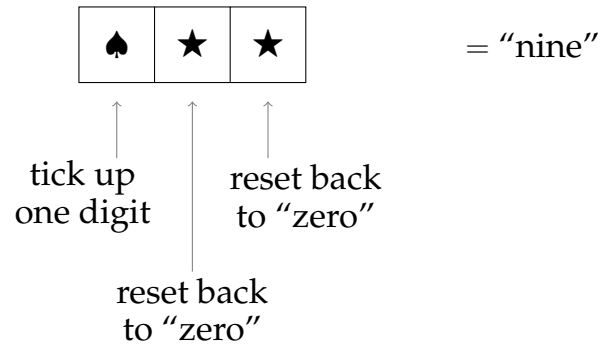
Now we can keep that second slot fixed, and we can cycle through the digits in our first slot again. So, for the next number (seven), we tick up the first slot:



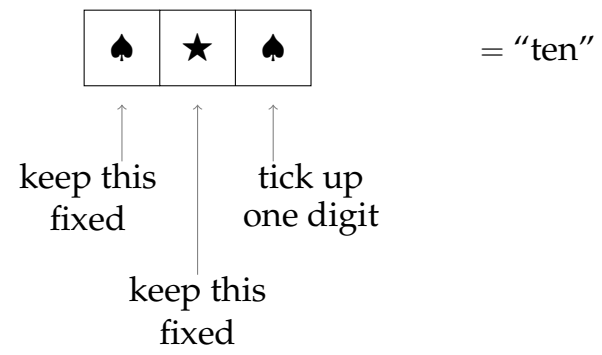
Then for the next number (eight), we keep the second slot fixed and tick up the first slot:



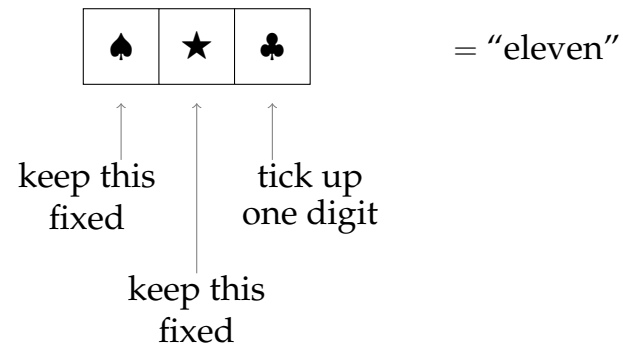
How do we write the next number (which is nine)? We've run out of digit symbols to use in the first slot, so we need to increment the slot to the left. But we've also run out of digit symbols to increment in the second slot too. So, we just move over to the next available slot. Hence, we reset the first two slots back to "zero," and then we tick up the third slot:



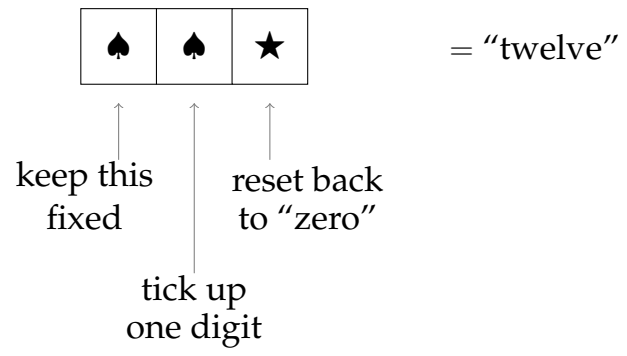
To do the next number (the number ten), we keep the two slots on the left fixed, and we tick up the first slot on the right:



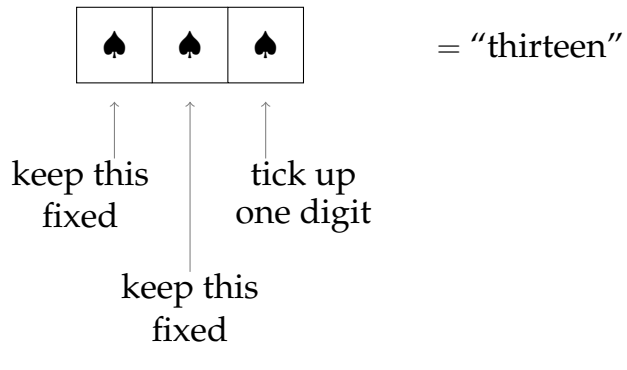
To get to the next number (eleven), we again keep the two slots on the left fixed, and we tick up the first slot on the right:



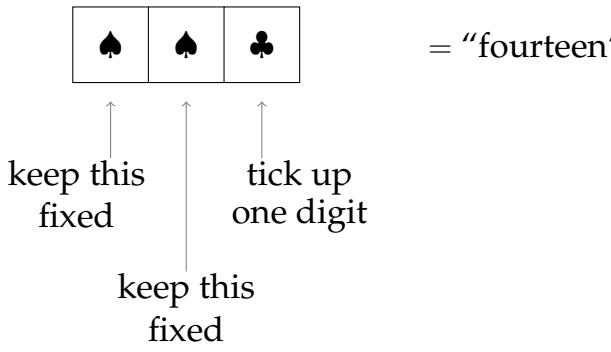
How do we do the next number (twelve)? We've used up the digits in the first slot, so we reset it to "zero," and we tick up the second slot one digit:



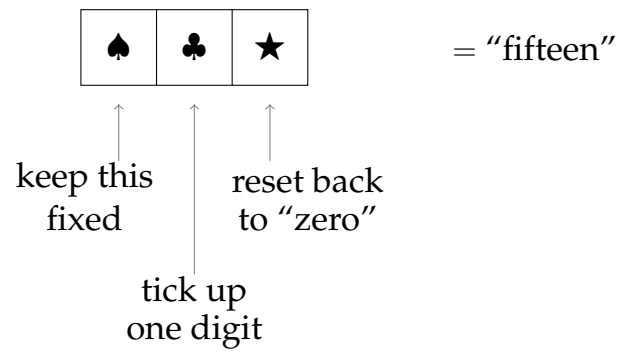
To get thirteen, we keep the left two slots fixed and we tick up the slot on the right:



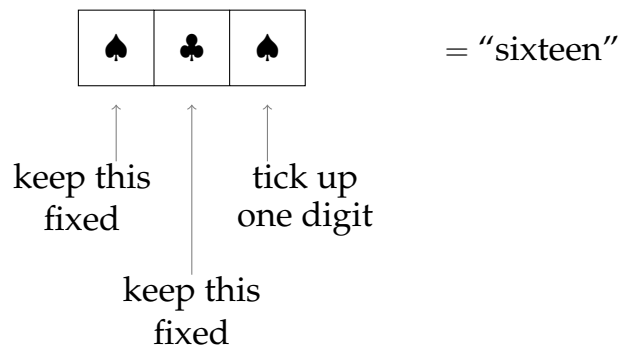
To get fourteen, we do the same thing again. We keep the left two slots fixed, and we tick up the slot on the right:



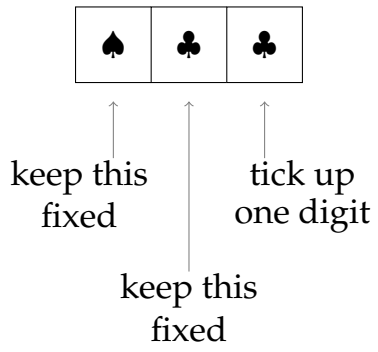
How do we get fifteen? We reset our first slot (the far right) back to "zero," then we tick up the second slot:



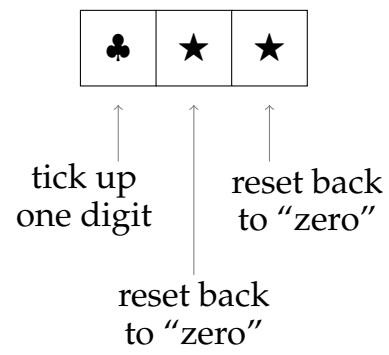
To get sixteen now, we just tick up the digit on the far right:



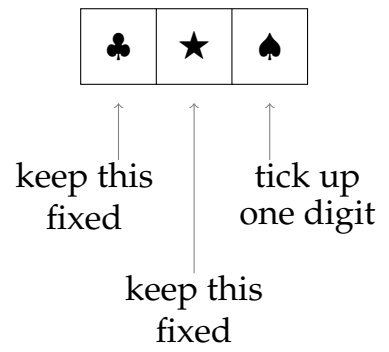
For seventeen, we tick up the right digit again:



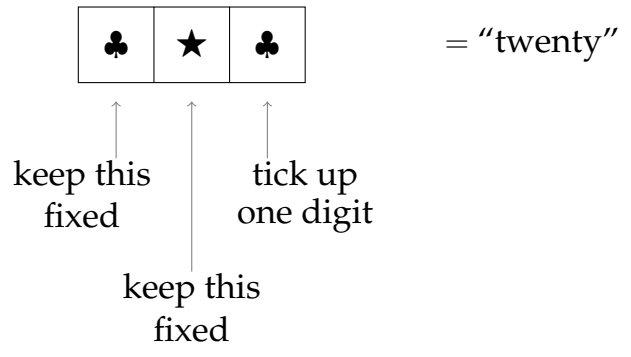
How about eighteen? We've used up our digits in the first and second slot, so we need to reset them back to "zero" and tick up the far left slot:



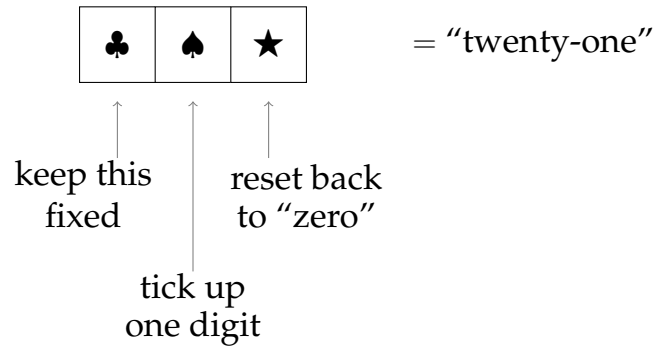
Now we can increment the 1st slot again. To get nineteen:



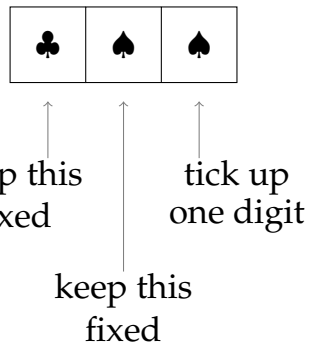
To get twenty:



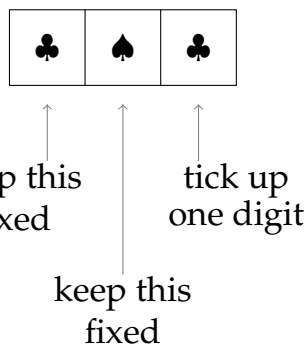
We've run out of digits on the far right side, so to get the next number (twenty-one), we need to reset the 1st slot back to "zero" and tick up the second slot:



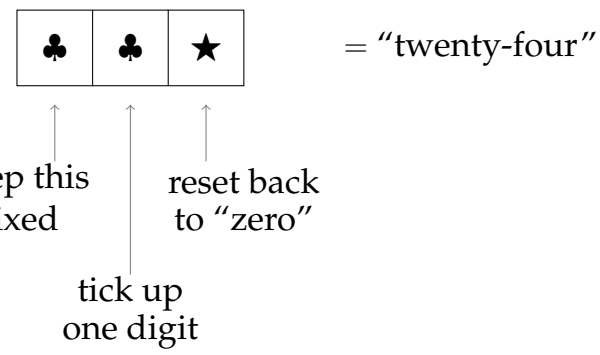
To get the next number (twenty-two), we can tick up the first slot:



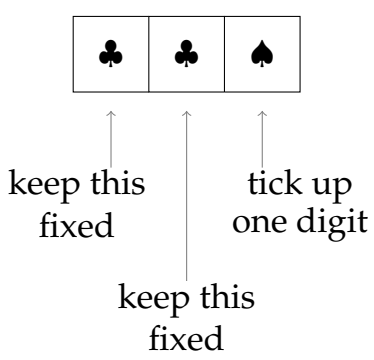
To get the twenty-three, we can tick up the first slot again:



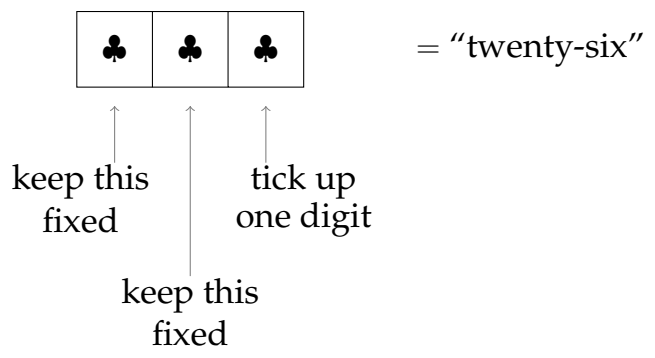
How do we get twenty-four? We've run out of digits in the first slot, so we need to reset it back to "zero" and tick up the second slot:



To get twenty-five, we can tick up the first slot:



And to get twenty-six, we can tick up the first slot one more time:



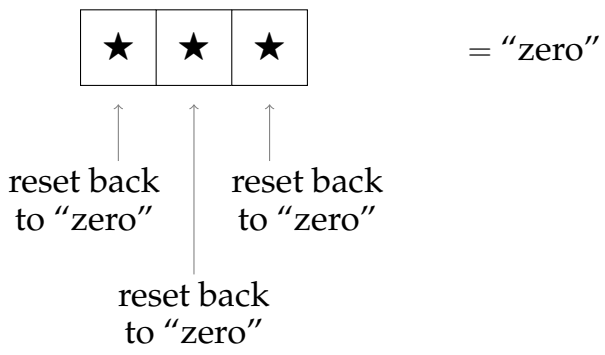
You can see how the pattern goes. To make bigger and bigger numbers, we tick up the far right slot, one digit at a time, until we run out of digits. When we run out of digits, we reset that far right slot back to "zero," and we tick up the slot to the left. Then we can cycle through the digits in the first slot again. We keep repeating this process, to get higher and higher numbers. Whenever we run out of digits in a slot on the right, we just tick up a slot on the left, to get more numbers.

55.3 Cyclic Numbers

CAN WE USE THE SYSTEM we devised in the last section to make a higher number? For instance, can we encode the next number (twenty-seven) with our digit symbols and our three slots?

The answer is: no, we cannot, with only three slots. In the last section, we incremented each of our three slots to their highest possible digits, and we were able to get up to twenty-six. But if we want to get higher, we need a new slot. To get to a bigger number, we'd need add a fourth slot on the left side. Then we could reset our slots back to "zero," at which point we could start incrementing the far right slot again.

This leads to an important point. Suppose that we *can't* add a fourth slot. For whatever reason, suppose that we only have three slots, and that's it. What happens if we reset the digits back to "zero"? What number do we get? We get zero:



So, with three slots and three digit symbols, we can only count up to twenty-six! If we keep cycling our digits after we hit twenty-six, we'll end up resetting our counter back to zero, and start counting upwards again.

If we have a fixed number of slots, we have something like a clock. On a clock, we start at the number one, then we count

up to the number twelve, then we start over at again at the number one. In our case, it's just the same, except we start at the number zero, then we go up to the number twenty-six, and then we start over at zero again.

So, if you have a fixed number of slots and can't add new slots on the left when you need to make bigger numbers, you will have clock counting (clock arithmetic). But if you can keep adding new slots on the left side whenever you need, then you can keep getting bigger and bigger numbers, and you won't have clock counting.

55.4 Summary

IN THIS CHAPTER, we looked at a way to represent numbers, by picking a small set of digit symbols and then cycling through them in slots.

- We start by incrementing the digits on the far right, and whenever we run out of digits in the far right slot, we reset the slot back to "zero" and we increment the next slot to the left.
- In this way, we can get bigger and bigger and bigger numbers with only a few digit symbols.
- However, we also saw that, if we can't add more slots, then we end up with a clock-counting (clock arithmetic) system, because at a certain point, our counting will reset back to zero.

[56]

COMMON NUMBER ENCODINGS

IN THE LAST CHAPTER, we looked at a way to encode or represent numbers by using a small set of digit symbols that we placed into slots. In this chapter, we will look at a family of different number encoding systems that are all based on that same idea.

56.1 The Octal System

IN THE LAST CHAPTER, we constructed a way to represent numbers using three digit symbols. We can call that system a **ternary** number encoding system, because "ternary" means "three" and we used only three digit symbols to write down our numbers.

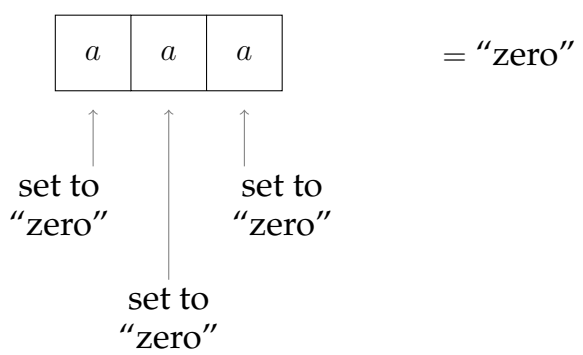
However, there's no reason that we have to use three digit symbols. We can build the same kind of encoding scheme using (say) eight digit symbols. We can call such an encoding scheme an **octal** encoding system.

So let's build such an octal system. Let's suppose that our eight digit symbols are the following:

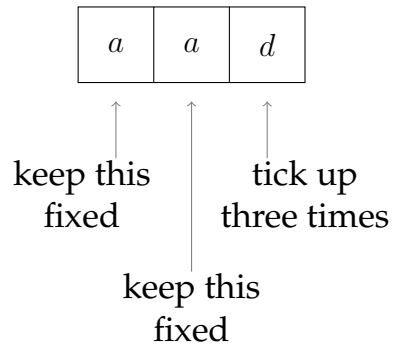
$$a, b, c, d, e, f, g, h$$

Let's say that these are in order, so "a" is the "zero" digit, and then "b" is the next digit up, then "c" is the next digit up, and so on, until we get to "h." The "h" is the highest digit we can get to, so when we reach "h," we need to tick up the next slot to the left to get to a bigger number.

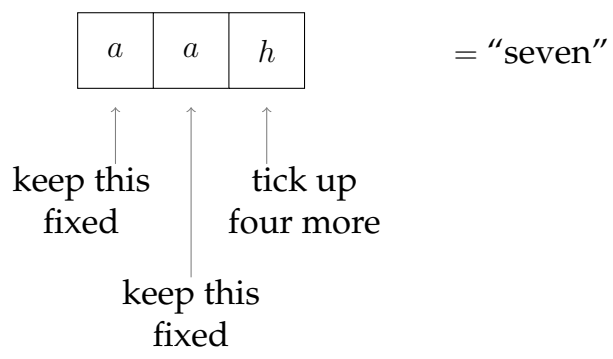
To write "zero," just like before, we set all of our slots to the "zero" symbol, which in this system is "a":



To write the number "three" in this octal sytsem, we tick up the first slot three digits:

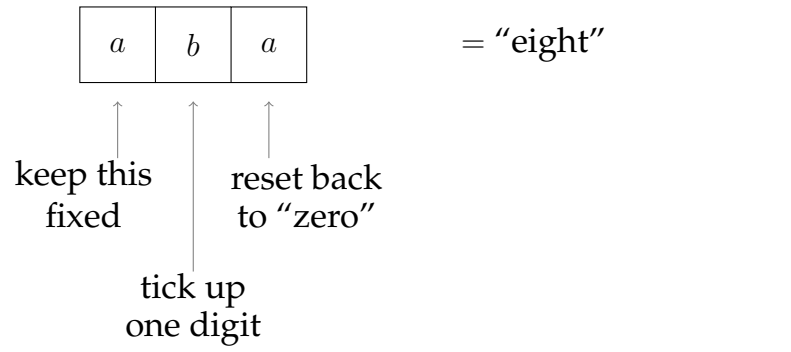


To write "seven" we tick up the first digit four more times:



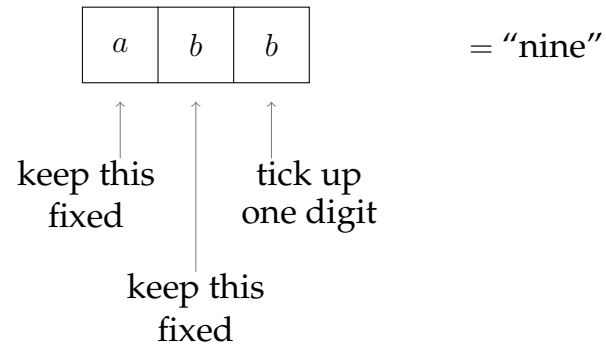
So, we write the number "zero" as "aaa," we write the number "three" as "aad," and we write the number "seven" as "aah."

Now let's do the number "eight." How do we write this in our octal system? We've run out of digits in the first slot, so we need to reset the first slot back to "zero" and tick up the next slot to the left:

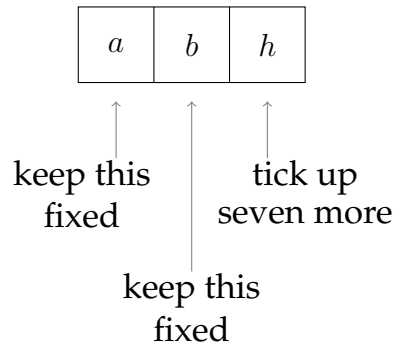


So to write the number "eight" in this octal system, we write "aba." Notice that we were able to count up to seven before we needed to reset our first slot. This is because we started with eight digit symbols instead of three. Since we have more digit symbols to use, we can count higher before we have to reset our first slot and tick up the next slot to the left.

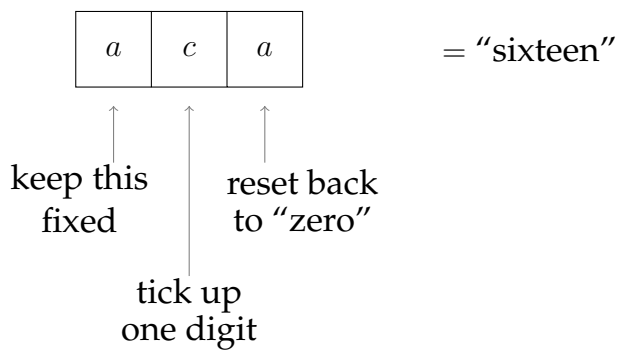
To get the number nine in our octal system, we can now just tick up the first symbol one digit:



To get to fifteen, we can tick up the first slot seven more times:



So, to write "fifteen" in this octal system, we write "abh." How do we get to "sixteen"? We are out of digits in the first slot, so we need to reset the first slot back to "zero" and then tick up the left slot one digit:



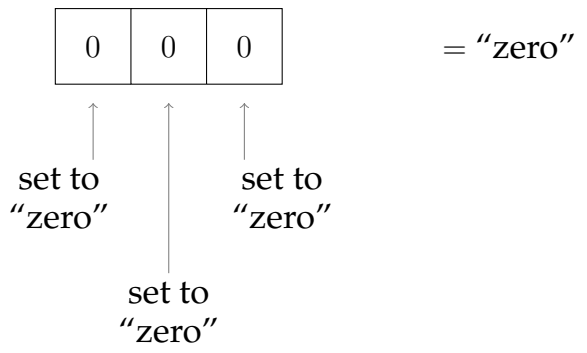
You can see how we are doing the same thing that we did with our ternary system, except that we have eight digits to cycle through before we tick up the next slot to the left.

56.2 The Decimal System

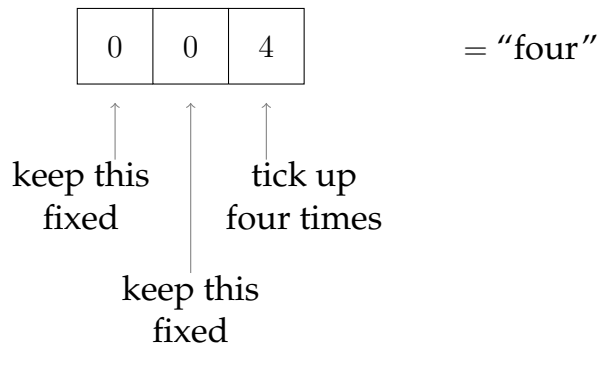
WE HAVE SEEN TERNARY AND OCTAL number representation systems. Let's look at a decimal system. We call it **decimal** because it uses ten digit symbols. Here they are, in order:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$$

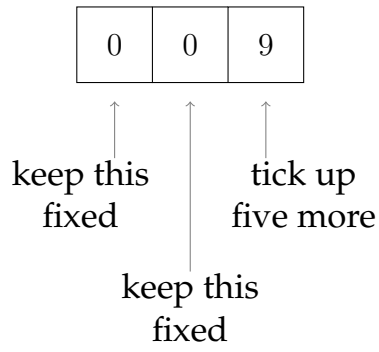
Let's write the number "zero." To do this, we set each slot to the "zero" digit symbol, which in this system is "0":



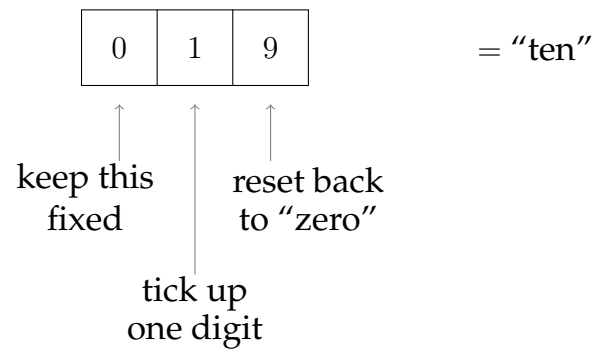
To get to the number "four," we tick up the first slot four times:



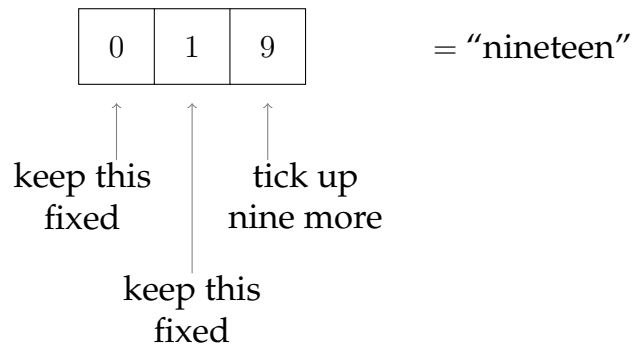
To get to "nine," we tick up the first slot five more times:



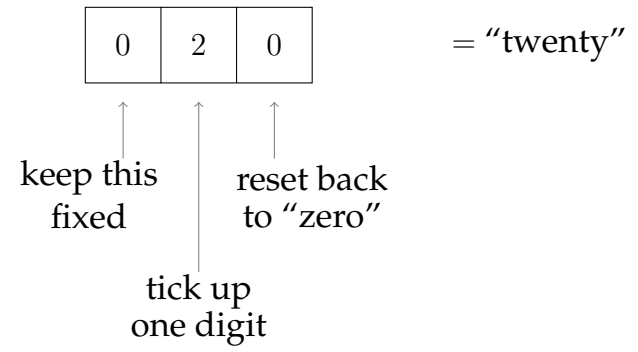
How do we get to the next number (which is ten)? We’ve run out of digit symbols to cycle through in the right slot, so we reset the right slot back to “zero” and we tick up the next slot to the left:



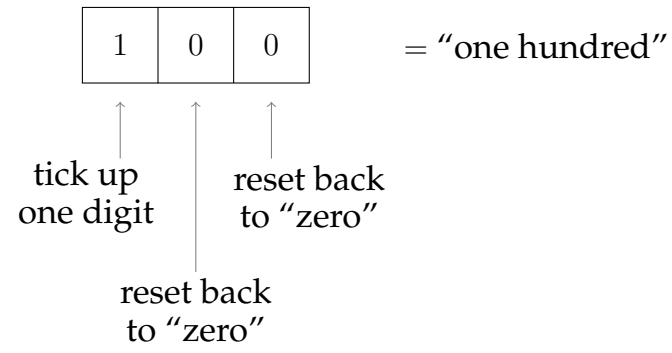
To get to nineteen, we can now tick up the first slot nine times:



How do we get to twenty? We have run out of digits to cycle through on the right side, so we reset the first slot back to zero and tick up the next slot on the left:



When we get to ninety-nine, how do we get to the next number (one hundred)? We reset the first two slots back to zero, and we tick up the next slot on the left:



This decimal system is precisely the number system that most people use when we write numbers down. Of course, we usually don't write the zeros on the far left. For example, when we write the number five, we don't write "005." We drop the zeros on the left, and just write "5." Likewise, when we write the number ten, we don't write "010." We drop the zero on the left, and just write "10."

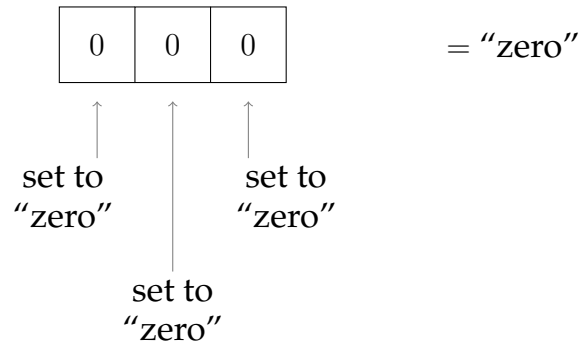
Still, you can see how this way of writing numbers (this **decimal** system) is no different than the ternary and octal systems we looked at before, except for the number of digits we have to cycle through. With this decimal system, we cycle through ten digits (zero through nine) before we tick up the next slot to the left. With the octal system, we cycle through eight digits before we tick up the next slot to the left, and with the ternary system, we cycle through three digits before we tick up the next slot to the left.

56.3 The Hexadecimal System

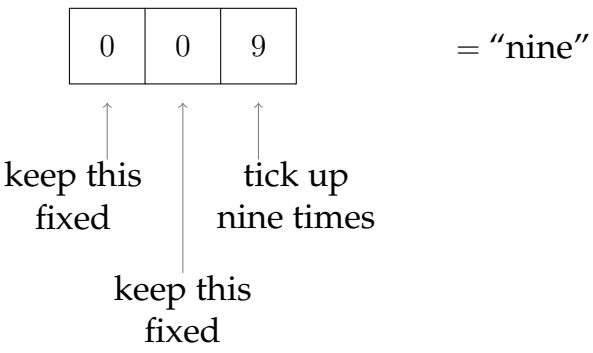
THERE IS ANOTHER SYSTEM that is often useful for talking about really big numbers. It is the **hexadecimal** number representation system. Like the ternary, octal, and decimal systems, this one works the same way, but we have *sixteen* digit symbols that we cycle through before we tick up the next slot to the left. Here are the digit symbols that hexadecimal systems use:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f$$

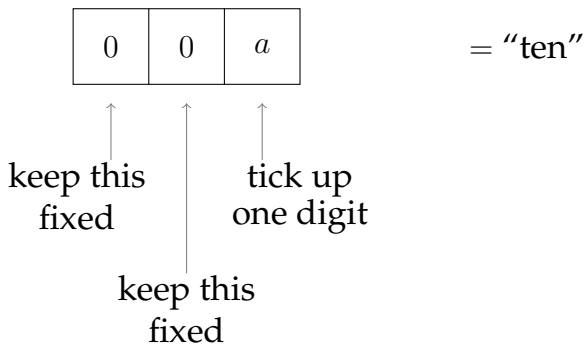
As you can see, there are sixteen symbols here. The first ten are the same digit symbols used by the decimal system. Then, for the next six symbols, we just use letters from the alphabet. To write "zero" with this hexademical system, we set all of our slots to the "zero" character (which in this system is "0"):



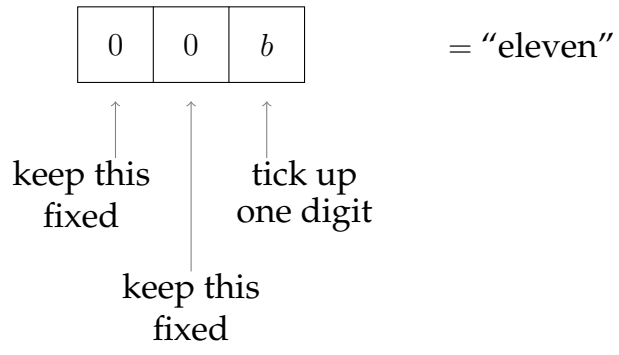
To count up to 9, we tick up the right slot nine times:



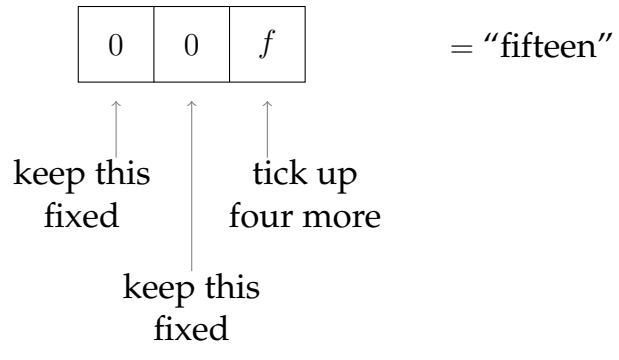
To get to the number ten, we tick up the right slot one more digit. The next digit symbol available from our list of digit symbols is "a," so this is how we write "ten":



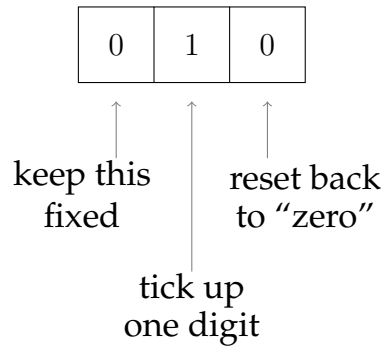
To get "eleven," we tick up the right slot one more time:



And to get up to fifteen, we tick up the right slot four more times:



How do get to sixteen? We have used up all of our digit symbols in the right slot, so we need to reset the far right slot back to "zero," and we tick up the next slot to the left:



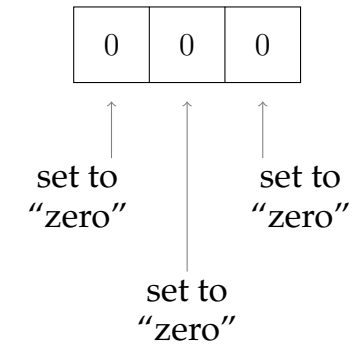
So, to write sixteen in the *hexadecimal* system, we write "010." Notice that this looks exactly like how we write ten in the *decimal* system. When we work with different number representation systems, it is important that we be clear about which representation system we're using. Otherwise, we might mistake "010" for ten, when really it was meant to be sixteen.

The hexadecimal system is convenient when we work with big numbers. This is because we have sixteen digits that we can cycle through (rather than only ten in the decimal system), so we can represent bigger numbers with fewer slots.

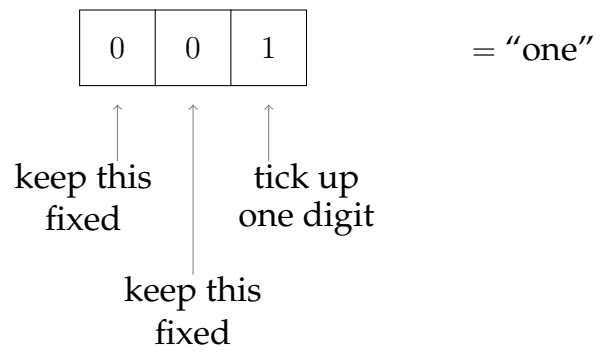
56.4 The Binary System

THERE IS ALSO PERHAPS THE SIMPLEST SYSTEM for representing numbers, which is the **binary** system. For this, we have only two digit symbols! Usually, we represent them with a "0" and "1" but they could be any symbols really.

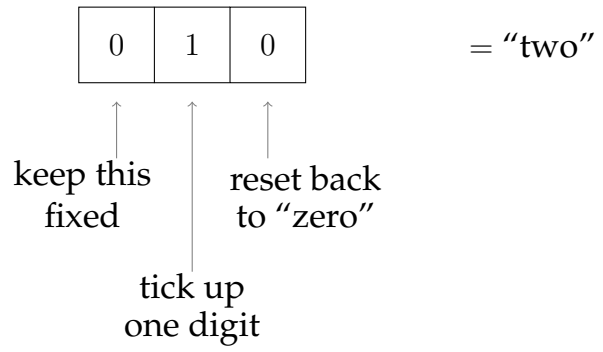
To write the number zero in our binary encoding system, we fill all the slots with the "zero":



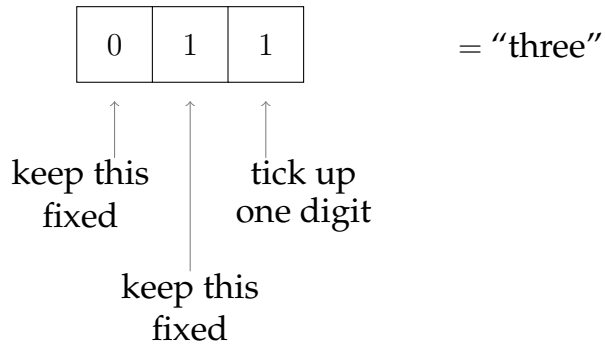
To get to the number one, tick up the far right slot:



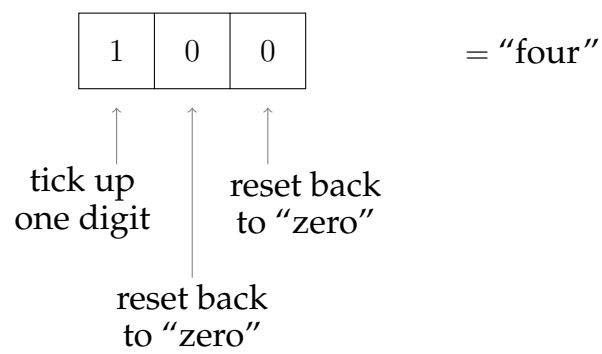
How do we get to two? We've already used up our digits in the right slot, so we need to reset the far right slot back to "zero," and then tick up the slot to the left:



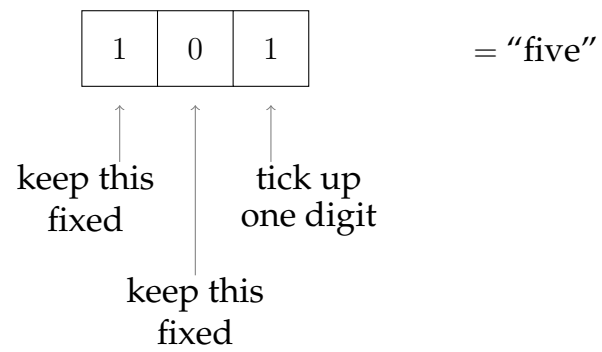
To get to three, we can then tick up the right slot one digit:



To get to four, we've used up our digits, so we need to reset our first and second slot back to zero, and tick up the third slot:



To get to five, we can then tick up the first slot one digit:



And so on. You can see how the pattern goes. (What is the biggest number we can make with the binary system and only three slots? It is seven: 111. See if you can work it out on paper.)

You can see from this that because we have such a small number of digit symbols available to us (we only have two digits available to us), we will need many more slots to encode bigger numbers.

So the binary system is a bit like the opposite of the hexadecimal system. With the hexadecimal system, we use more digits and fewer slots, whereas with the binary system, we use fewer digits and more slots.

56.5 Summary

IN THIS CHAPTER, we looked at some common number encoding schemes that are all based on the idea of using a small set of digit symbols that we cycle through in slots. In particular, we looked at the **octal** system, the **decimal** system, the **hexadecimal** system, and the **binary** system.

[57]

THE PARTS OF THE CPU

IN THIS CHAPTER, we will look at the basic parts that go into the CPU (Central Processing Unit) of a computer, so that we can understand how it actually works. To keep things simple, we will put together a very basic CPU, but all modern computers are essentially just beefed up versions of the same thing.

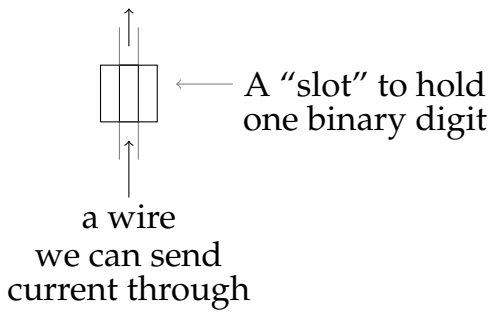
57.1 Numbers "in" the Computer

PREVIOUSLY, WE LOOKED AT A VARIETY of ways to encode numbers. For instance, we looked at the decimal system (which is what most of us use daily), we looked at the hexadecimal system, and we looked at the binary system.

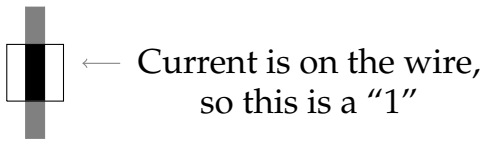
How might we realize any of these number systems inside a computer? How might we "write down" a number, inside a computer, so that a computer can do something with that number (like add it to another number)? There isn't a piece of paper inside the computer, so we can't write it down directly.

But there are wires inside the computer, and we can use wires to represent a number. A wire can have some amount of current flowing through it. So, we can say that if a wire has no current flowing through it, then it represents a "0," and if it has current flowing through it, then it represents a "1."

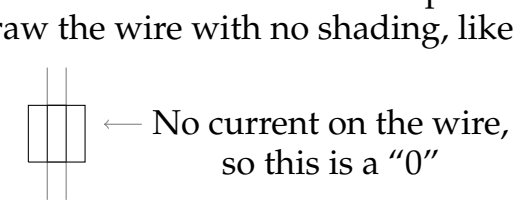
So, we have a way to represent a binary digit! Let's imagine that we make a little box, that has a wire running through it, like this:



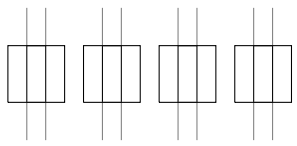
To represent that current is passing through the wire, let's draw the wire shaded, like this:



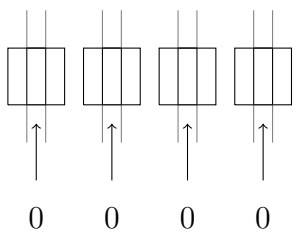
To represent that there is no current passing through the wire, let's draw the wire with no shading, like this:



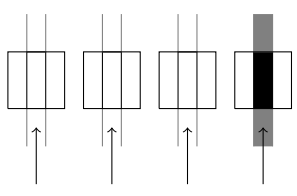
So, what we have here is a little component that can represent a binary digit, and it uses nothing but electronics. Next, let's imagine putting together an array of these little components, so we can have more than one digit. Let's imagine four components, side by side:



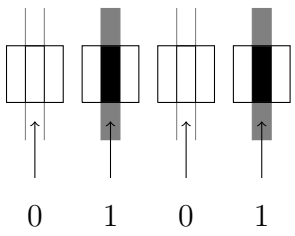
Each of these components represents one digit, which we call a "bit" of information. Since we have four components here, this can represent a 4-bit number. For instance, it represents the number zero (in binary: 0000) when all of the wires have no current running through them:



It represents the number one (in binary: 0001) when there is current going through the wire on the far right:



It represents the number 5 (in binary: 0101) when the second and fourth wire has current going through it:



0 1 0 1

So this set of four components can represent any 4-digit number, in binary.

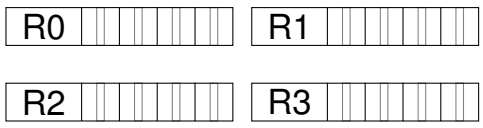
57.2 Scratch registers

THE COMPONENT we built in the last section can hold any 4-bit number, where the number is encoded as a binary number.

We can build a number of these little components, and put them inside of a computer. That way, when the computer needs to do some scratch work (calculations with numbers), it can use our little components as places where it can "write down" the numbers it needs to work with.

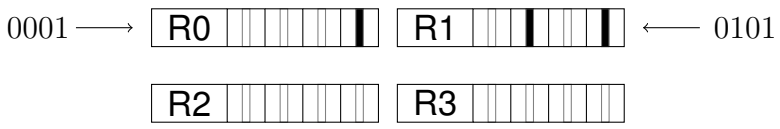
We call a component that can hold a number for a computer's scratch work a **scratch register**. The scratch register we are imagining here is a **4-bit** scratch register, because it can hold a 4-bit number.

Every modern computer comes with at least a few dozen scratch registers. Each scratch register has a special name too. Let's imagine that we have four scratch registers, named $R0$ (for "Register 0"), $R1$ (for "Register 1"), and so on. Like this:



Remark 57.1. Most computers today have 32-bit or 64-bit scratch registers. In other words, each scratch register has 32 or 64 "slots," each capable of holding a "zero" or a "one." For the sake of simplicity, we will stick to 4-bit registers, but the only difference between a 4-bit register and a 32- or 64-bit register is that the bigger register can hold bigger numbers, since it has more "slots" to write down digits in.

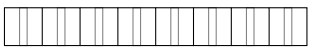
For example, the computer can "write down" the number one (in binary: 0001) in scratch register $R0$, and it can "write down" the number five (in binary: 0101) in scratch register $R1$:



57.3 Memory

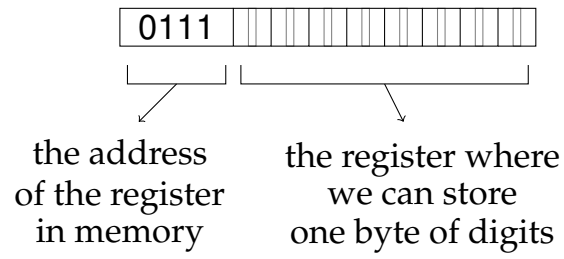
IF OUR COMPUTER NEEDS MORE PLACES to write things down (when the scratch registers aren't enough), let's also imagine that it will have a stack of extra registers over on the side. We call this stack of side registers the **memory** of the computer (or to be more exact, its **Random Access Memory**, also known as **RAM**).

Let's suppose that these side registers are each 8-bits in size. That is, each one can hold 8 binary digits. So, whereas our scratch registers were 4-bit registers (because they could hold 4 binary digits), these registers are 8-bits in size (because they can hold 8 binary digits). The size "8-bits" is used so often we have a special name for it: we call it a **byte**. So each of these side registers can hold one *byte* of digits. To draw a picture, one of these side registers would look like this:

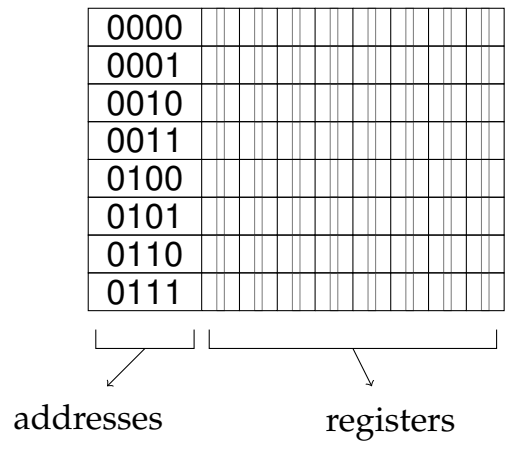


Let's also say that each side register is labeled with a number. We'll call this the **address** of the side register. It's kind of like the address to a house. When we are looking for a particular house on a street, we look for its house number. Likewise here. This side register has an address, and we can

pick out this particular side register by its number. Here's a picture, with the address attached to it:

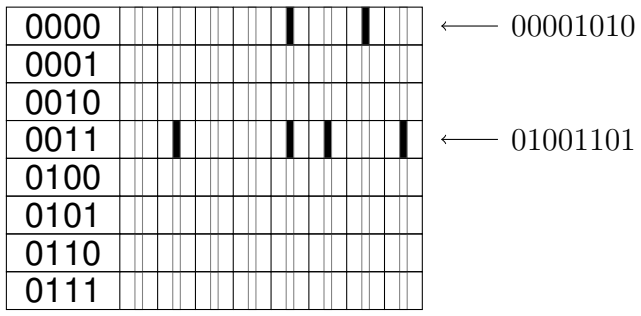


Next, let's imagine that we stack a number of these addressed side registers on top of each other. For our purposes here, let's suppose that we have just eight side registers, stacked on top of each other. Let's say that the register on the top of the stack has address "0" (in binary: 0000), the next one underneath it has address "1" (in binary: 0001), then the next has address "2" (in binary: 0010), and so on. Like this:



In essence, this gives us a memory bank. It is a stack of "slots" (side registers) that we can put numbers in, where each "slot" has an address. For instance, we can put the number five (in binary: 00001010) in the slot addressed as

"0000," and we can put the number seventy-seven (in binary: 01001101) in the slot addressed as "0011":



Since this stack has eight slots, and each slot holds one byte (8-bits), this memory bank can hold a total of 8 bytes (or 64-bits). In computer speak, we can say that our imaginary machine here has 8-bytes of **memory** (or synonymously, we can say it has 8-bytes of **RAM**).

Remark 57.2. Most modern computers have many more slots than we have here. Our imaginary machine only has eight slots, but that is only because we want to keep things simple, for pedagogical reasons.

57.4 The ALU

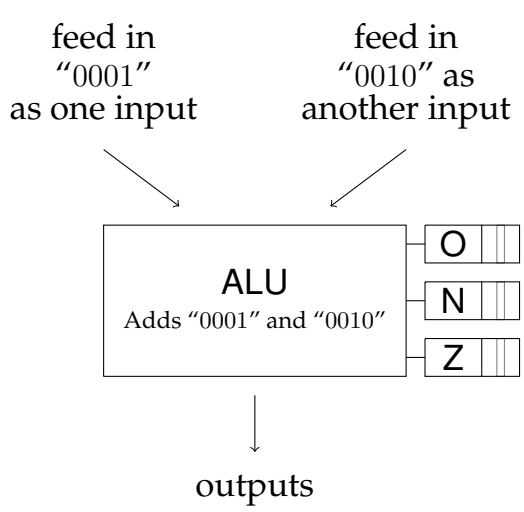
WHAT ABOUT DOING ARITHMETIC AND BASIC COMPUTATION? For this, every computer has a special little chip inside it called the **ALU** (short for the "Arithmetic and Logic Unit"). There is no special magic to the ALU. It is just a bunch of little circuits that let us do some *extremely basic* arithmetic with binary numbers.

Let's draw the ALU as a little box, like this:

The ALU also has attached to it three 1-bit registers. They are called "O," "N," and "Z." Let's draw them like this:

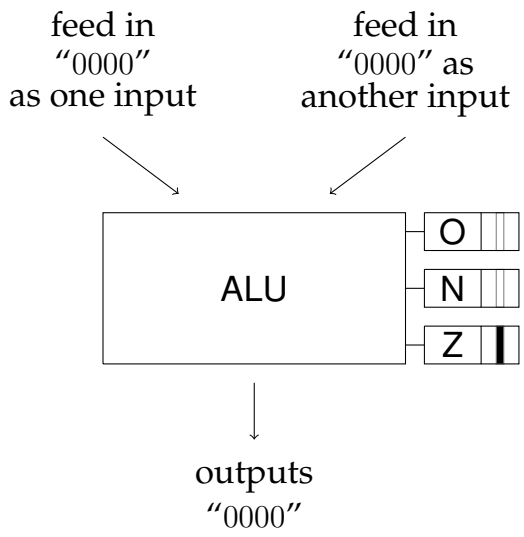
We will talk about what these little registers are for in a moment.

In the most basic terms, the ALU works like this: we put two binary numbers into it, it adds them together, and it spits out the sum as a new binary number. For instance, suppose we want it to add "one" (in binary: 0001) to "two" (in binary: 0010), and so produce "three" (in binary: 0011). We would feed the ALU "0001" and "0010" as inputs, it would add them together, and then spit out "0011." Something like this:



Let's talk about those little 1-bit registers now. The "Z" register is called the **Zero Flag**. When the ALU performs a

computation, if the result is a zero, it will turn on that "Z" bit. For instance, suppose we ask the ALU to add zero and zero. The result of zero plus zero is zero. So then, the ALU will spit out "0000" as its output, but it will also turn on its zero flag. So, it will look like this:

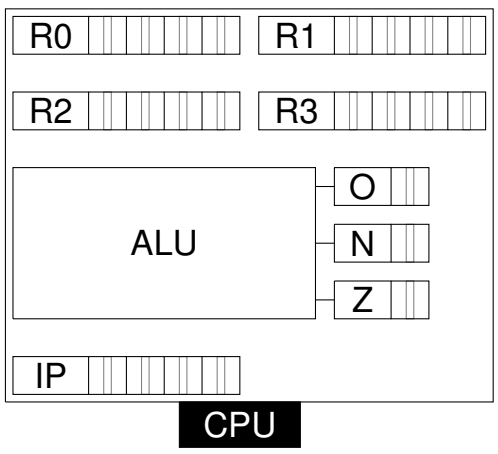


What about the "N" register? The ALU will turn this one on if the result is a negative number. We haven't talked about how to represent negative numbers inside the machine, but it can be done, and if the result of the ALU's computation is negative, it will turn on this "N" bit. We call the "N" register the **Negative Flag**.

What about the "O" flag? Suppose we try to add 1111 and 0010. What is the result? Well, if the ALU can only handle 4-bit numbers, it can't produce a number bigger than 1111. So, it will wrap around back to zero (like a clock), and start adding again from there. If the ALU wraps around like this, we call it an **overflow**. If an overflow happens during a computation, the ALU will turn on the "O" bit. Hence, we call the "O" register the **Overflow Flag**.

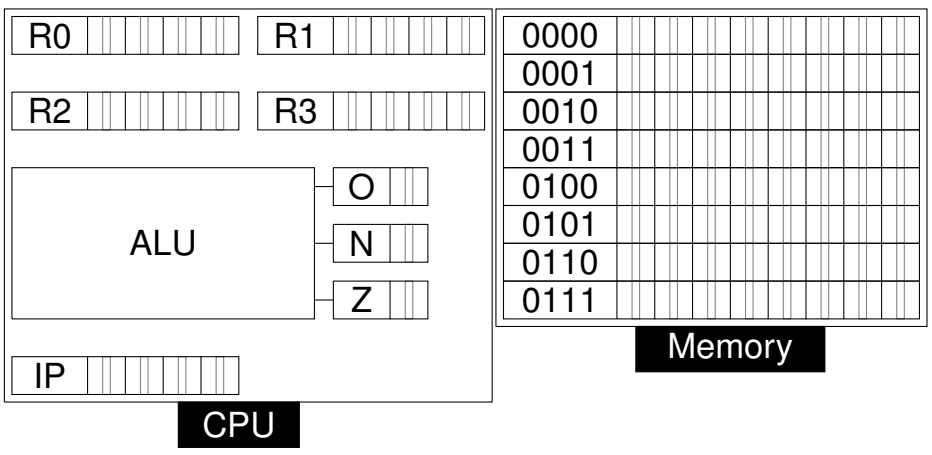
57.5 The CPU

AT THIS POINT, we have all the basic pieces we need to put together a **CPU** (short for the "Central Processing Unit," or just the **processor**). A processor contains some scratch registers and an ALU, so it can perform computations. It also has one more register, called an **Instruction Pointer**, or "IP" for short. This register holds an address to memory, but we'll talk more about what this does in the next chapter. For now, it is enough to just put together a picture of a CPU. Here it is:



You can see the four scratch registers $R0$, $R1$, $R2$, and $R3$; you can see the ALU with its overflow flag, negative flag, and zero flag; and you can see the instruction pointer (called IP). All together, that makes a CPU.

The final step in building a computer is to attach the memory (our RAM). So, the CPU and the memory together make up the computer's internals. Like this:



57.6 Summary

IN THIS CHAPTER, we looked at how to represent numbers inside a computer, we learned about scratch registers, memory (RAM), the Arithmetic and Logic Unit (the ALU), and the Central Processing Unit (the CPU).

[58]

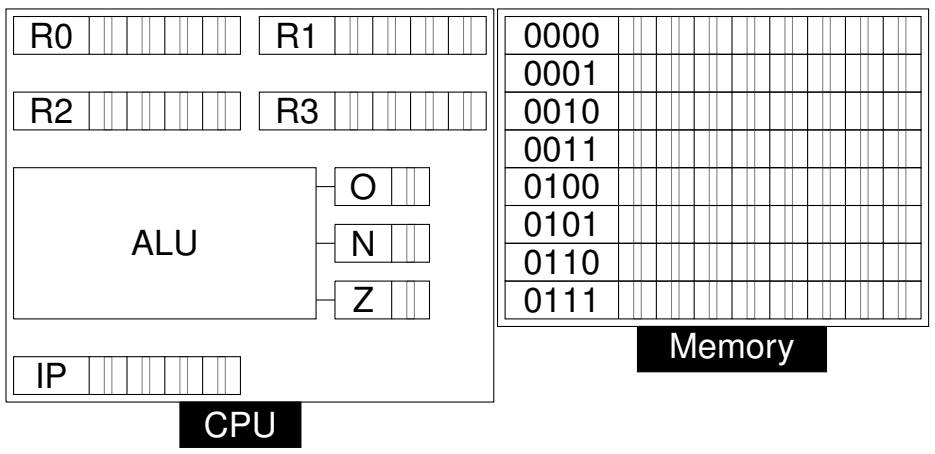
RUNNING PROGRAMS

IN THIS CHAPTER, we will look at how a CPU actually runs or “executes” computer programs. Again, we will keep things simple, but real computer programs are just beefed up versions of the same thing.

58.1 Machine Instructions

EVERY CPU is pre-built to understand and execute a small set of instructions. Each instruction is just a sequence of ones and zeros, that the CPU has been pre-built to recognize.

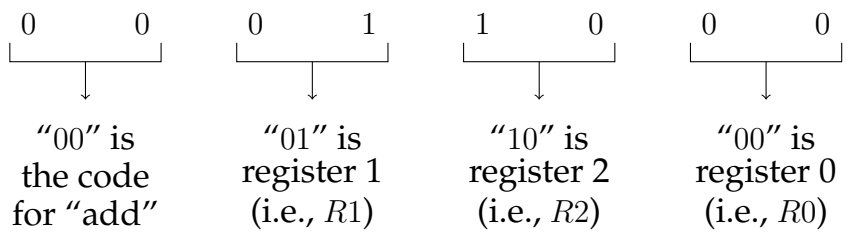
For example, let’s take the simple CPU we put together in the last chapter. Recall that it looks like this:



Let's suppose that this little CPU has been built to recognize the following 8-bit sequence of ones and zeros:

0 0 0 1 1 0 0 0

Our CPU interprets this code like so:



Our CPU parses this code by first looking at the initial two digits. If they are 00, then it says, "hey, I've been pre-built to recognize '00' as the code for 'add'." Then, once it knows it's going to be adding, it looks at the next two digits to find out which register it should look in for the first number to add. In this case, "01" is just the number "one" (in binary), and so that tells our CPU that it should look in register 1 (i.e., $R1$). Next, our CPU looks at the next two digits of the code to find out which register it should look in to get the next number to

add. In this case, "10" is the number "two" (in binary), and so that tells our CPU that it should look in register 2 (i.e., $R2$) for the second number to add. Finally, our CPU looks at the last two digits of the code to find out which register it should put the result in. In this case, "00" is the number zero, and so it that tells our CPU that it should put the result in register 0 (i.e., $R0$).

So, this code instructs our CPU to take whatever number is in register 1 ($R1$) and whatever other number is in register 2 ($R2$), then to use the ALU to add those two numbers together (setting the zero, negative, and overflow flags as appropriate), and finally, to put the result in register 0 ($R0$).

We might translate this code into a more human readable form, so that it says something like this:

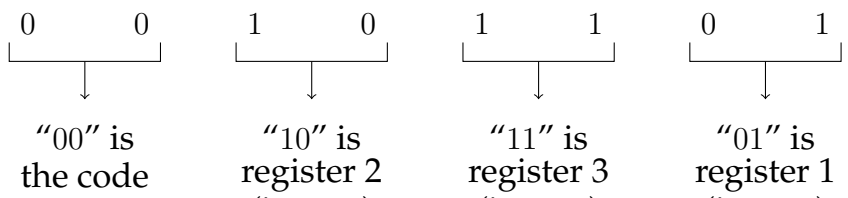
```
ADD     R1      R2      R0
```

That says the same thing as the binary-coded version earlier: it instructs the CPU to take whatever number is in register $R1$ and whatever number is in $R2$, then add those two numbers together (setting the zero, negative, and overflow flags as appropriate), and then put the result in $R0$.

If we want to tell our CPU to add the numbers from R2 and R3, and then put the result in R1, we would follow the same pattern and write a coded instruction that looks like this:

0 0 1 0 1 1 0 1

Our CPU would interpret that code like so:



Remark 58.1. Every CPU manufacturer builds their CPU to understand different coded instructions, which the manufacturer makes up. There is no "standard" set of instructions that every CPU expects, though of course many are very similar. Basically, the coded instructions you give to a particular CPU are like secret codes two spies give to each other. They agreed before hand on what the symbols will mean. So also with CPUs. The manufacturer agreed before hand what the sequences of ones and zeros will mean, and then they build their CPU to execute those instructions.

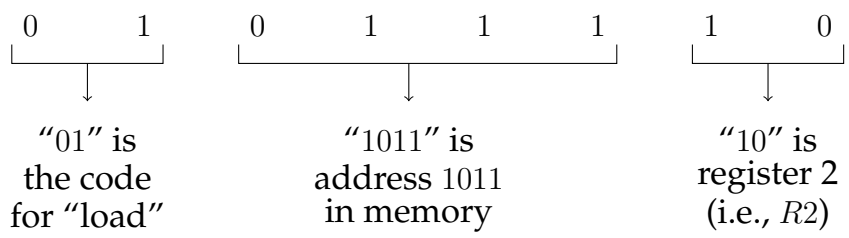
To translate that into our more human-readable form, we could write it like this:

```
ADD     R2      R3      R1
```

Next, let us suppose that our CPU has been built to recognize another kind of instruction, which looks like this:

0 1 0 1 1 1 1 0

Our CPU interprets that code like so:



To parse this code, our CPU again looks at the first two digits in the number. This time, they are "01" (rather than "00" as before). So, our CPU says, "hey, I've been pre-built to recognize '01' as the code for 'load' (i.e, pull info out of the memory bank)." Then, once it knows it's going to be loading something from memory, it looks at the next four digits of the code to find the address of the slot in memory it should fetch. In this case, 1011 is an address in our memory bank, so that tells our CPU to go to the register at address 1011, and read whatever number is stored there. Finally, it looks at the last two digits of the code to find out which register it should put that number in. In this case, 10 is the number two, so that tells our CPU to take the number it has fetched from the memory bank and put it into $R2$.

We can translate the binary-coded instruction into a more human-readable form, that looks something like this:

Remark 58.2. You may notice that $R2$ only has space for 4 bits, yet we are trying to load a number that is 8 bits. What this means is that our CPU will only be able to put 4 of the bits it read into $R2$. This is okay, because our registers can only deal with 4 bits at a time anyway, so the extra information in the memory banks is just extra information that our CPU isn't powerful to use. But if we had a bigger budget (let's say), we could have designed our scratch registers to each hold 8-bits, in which case our CPU would then be able to load all 8-bits stored at any address in the memory bank.

```
LOAD    @1011      R2
```

This says just what the binary-coded version says: it instructs our CPU to load the number it finds at address 1011 into the register $R2$.

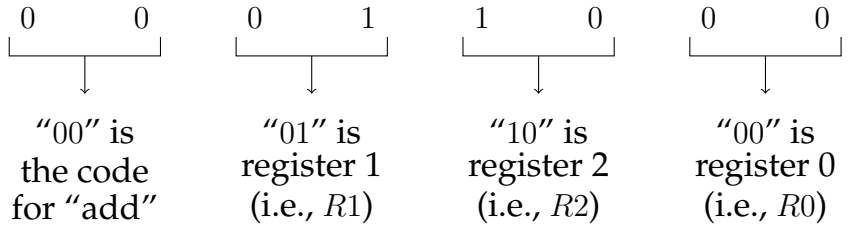
There are a few more instructions we could discuss, but you get the basic idea from this. Basically, a coded instruction is just a sequence of ones and zeros that our CPU has been pre-built to recognize.

58.2 Loading and Running a Program

LET US START with a very simple program. Let us suppose that we have the following two lines of code:

0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	1

What do these binary-coded instructions mean? Our CPU will interpret the first line like this:

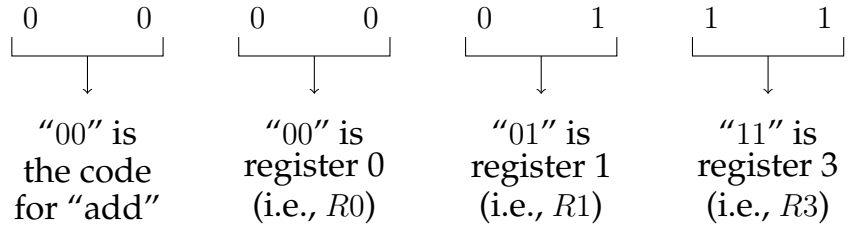


Which, in our more human-readable form, looks like this:

```
ADD       R1        R2        R0
```

In other words, this tells our CPU to add whatever numbers are in $R1$ and $R2$, and then put the result in $R0$.

As for the second line of our code, our CPU will interpret it like this:



Which, in our more human-readable form, looks like this:

```
ADD     R0     R1     R3
```

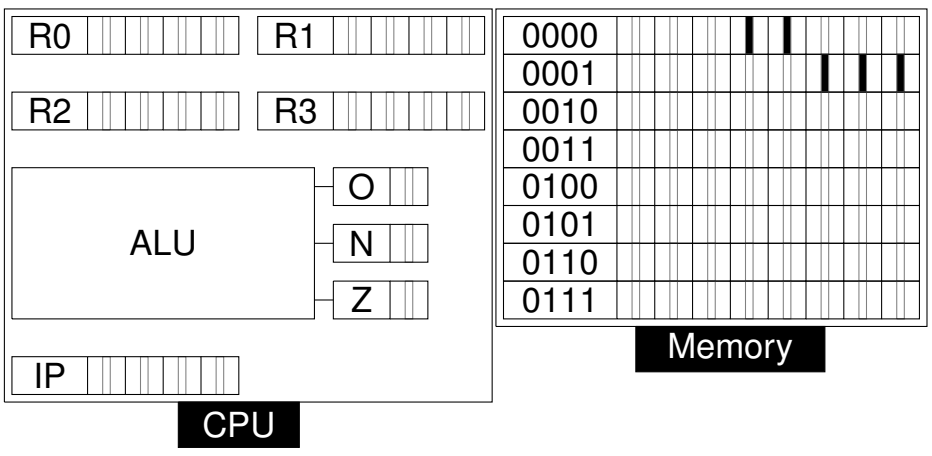
Terminology. The instruction coded in ones and zeros is called **machine code** because it is the actual code that the machine can interpret. The more human readable version of the code is called **assembly** code. As you can see, there is a bijective correspondence between the machine version and the assembly version (they are isomorphic). It is very easy to translate back and forth between them.

In other words, this line of code instructs our computer to add whatever numbers are in R0 and R1, and put the result in R3.

In human readable form, then, our two lines of code amount to this:

```
ADD  R1  R2  R0
ADD  R0  R1  R3
```

To get our program to run this code, we first have to load this code into our computer's memory. To do that, we put each line of our code into the memory bank. So, we put the first line of our binary-code (which is "00011000") into the register at the first address of memory (namely, address 0000), and then we put the second line of our binary-code (which is "00000111") into the register at the second address of memory (namely, address 0001). Then, our computer looks like this:



Notice how our two instructions occupy the first two slots in our memory bank.

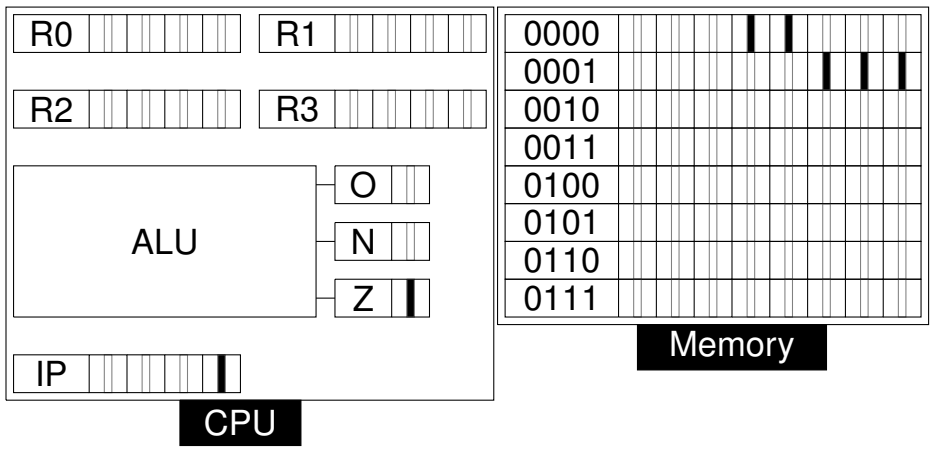
Now it is time to run the program. When we tell the computer to run the program, it first resets the **instruction pointer** (the register marked "IP" in the CPU) to "0000" (if it is not already at "0000").

The number in the IP register tells our CPU where to find the instruction of code it should execute. Since the IP register has the value "0000," that tells our CPU to look in memory at address "0000" for its first instruction.

So, our CPU goes to the memory at address "0000" and it sees the instruction "00011000." It pulls that out, and interprets it in the way we described before. Our CPU knows that this instruction tells it to take the numbers in $R1$ and $R2$ and add them together, then put the result in $R0$. So, our CPU does that. It looks in $R1$ and it sees that the number there is 0000, it looks in $R2$ and it sees that the number there is 0000, and then it feeds both of those numbers into the ALU. The ALU then adds those two numbers together. The result of adding zero and zero is zero, so the output is 0000. Since the result of the ALU is zero, it turns on the zero flag. Finally, the CPU takes the result (i.e., 0000) and puts it into $R0$ (which

already has 0000 in it, so we see no change).

At this point, our CPU has finished executing the instruction it found in memory at address 0000. So, it is finished with one round of execution, and it is time to move on. To finish, it increments the IP register by one, so that the CPU is ready to go to the next instruction in memory. Hence, after completing the first instruction, our computer looks like this:

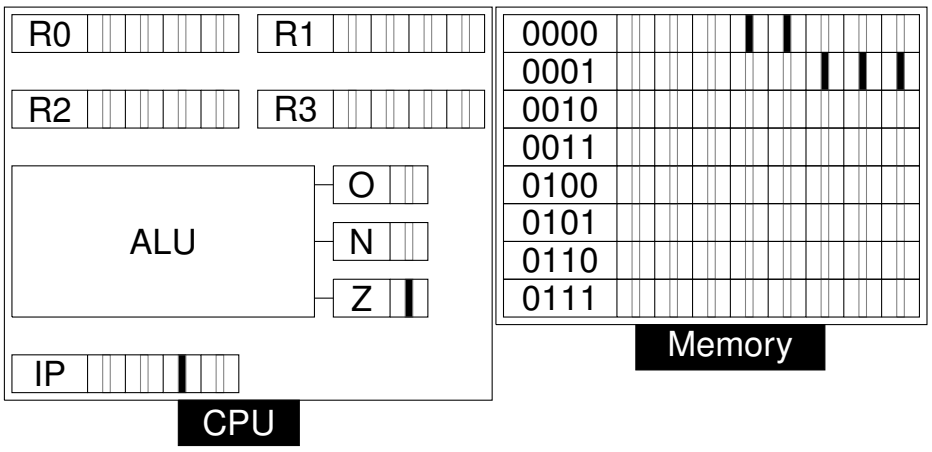


Notice that $R0$ contains the result of adding $R1$ and $R2$ — of course, it is just all zeros because the result of adding zero to zero is still zero. But nevertheless, the correct value is present in $R0$. Notice also that the zero flag is turned on, because the result of the ALU's computation was zero. Finally, notice that IP is at 0001, so that the CPU is ready for the next instruction.

Now the CPU begins round two. For this, it repeats just what it did for round one. It looks in IP for the address in memory where it can find the next instruction, it goes to that address, it executes the instruction there, and then it increments the IP register.

So, in this case, the IP register says to look at address 0001 for its next instruction, so it goes there to find 00000111. As we know, this tells our CPU to take the values in $R0$ and $R1$,

and put the result in $R3$. So, it does that. In this case, $R0$ and $R1$ contain zeros, and adding zero to zero results in zero, so the ALU turns on its zero flag (which was already on, so we don't see a change there), and then the CPU puts the resulting zero in $R3$ (which was already zero, so we don't see a change there). Then, the CPU increments the IP register by one, so that it reads 0010. Hence, after the second round of execution, our computer looks like this:



Note that $R3$ has zeros in it (which is the correct result of the arithmetic our program performed), the zero flag is on, and the instruction pointer has been incremented.

At this point, there are no more instructions in memory, so the computer has finished its execution, and we are done. Our program was simple, but our computer executed it, one instruction at a time.

Notice that this program always adds zeros, because all it does is add values stored in the registers, and the registers start with just zeros in them. If we want to add other numbers besides zeros, we need to first load some values into the registers. We will do that in our next program.

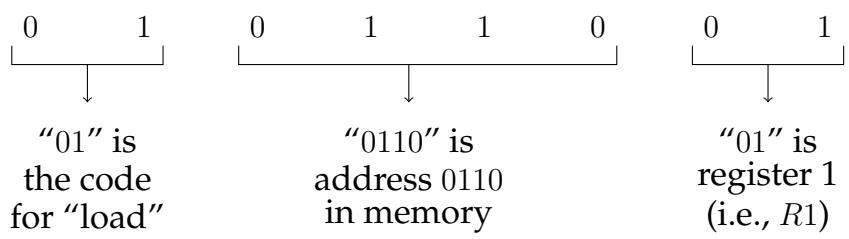
58.3 Starting with Values in Memory

LET'S LOOK AT ONE MORE PROGRAM. This one is similar to the last one, except we will put some custom values into our memory bank before we start up the program. Then our program can use those values when it runs.

For this second program, let us suppose that we have the following three lines of code:

0	1	0	1	1	0	0	1
0	1	0	1	1	1	1	0
0	0	0	1	1	0	0	0

What do these binary-coded instructions mean? Our CPU will interpret the first line like this:

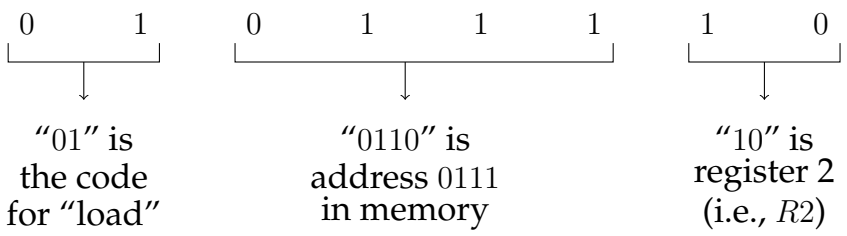


Which, in our more human-readable form, looks like this:

```
LOAD    @0110    R1
```

In other words, this tells our CPU to read whatever number is stored in memory at address 0110, and put it in the register $R1$.

As for the second line of our code, our CPU will interpret it like this:

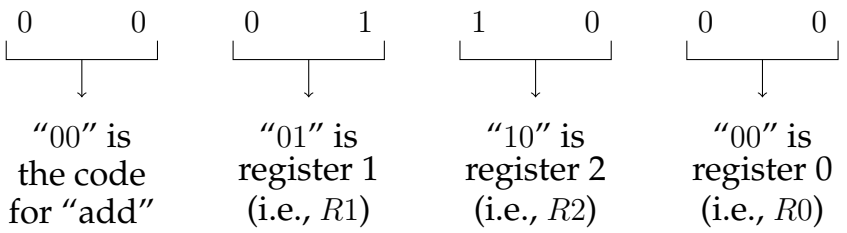


Which, in our more human-readable form, looks like this:

```
LOAD    @0111      R2
```

In other words, this tells our CPU to read whatever number is stored in memory at address 0110, and put it in the register $R2$.

Our CPU will interpret the third line of code like this:



Which, in our more human-readable form, looks like this:

```
ADD        R1         R2         R0
```

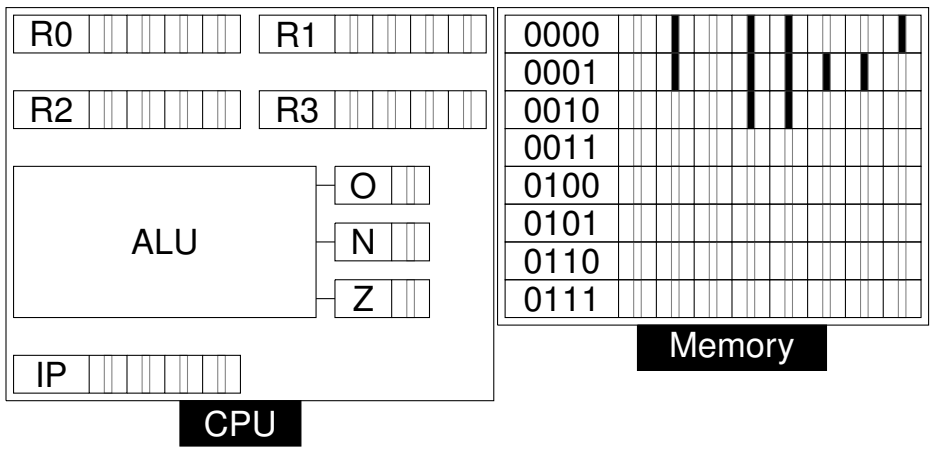
In other words, this tells our CPU to add whatever numbers are in $R1$ and $R2$, and then put the result in $R0$.

In human readable form, then, our three lines of code amount to this:

LOAD	@0110	R1	
LOAD	@0111	R2	
ADD	R1	R2	R0

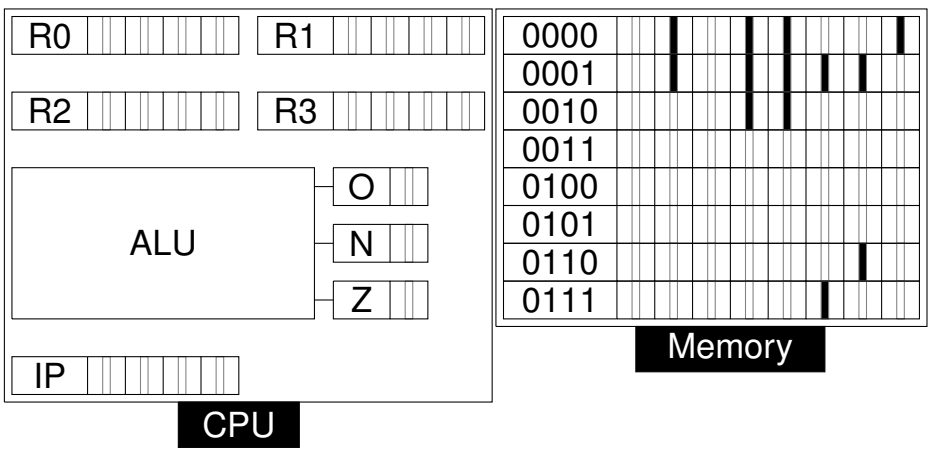
So, our program will first load the value from memory at 0110 into $R1$, then it will load the value from memory at 0111 into $R2$, then it will add $R1$ and $R2$ and put the result in $R0$. In effect, this is a little calculator. We put the two numbers we want to add into memory at addresses 0110 and 0111, then we run the program, and the result will be in register $R0$.

To get our program to run this code, we first have to load this code into our computer's memory. To do that, we put each line of our code into the memory bank. So, the first three lines of code go in the first three registers in our memory bank. After loading the code into memory, our computer looks like this:



Notice how our three coded instructions occupy the first three slots in our memory bank.

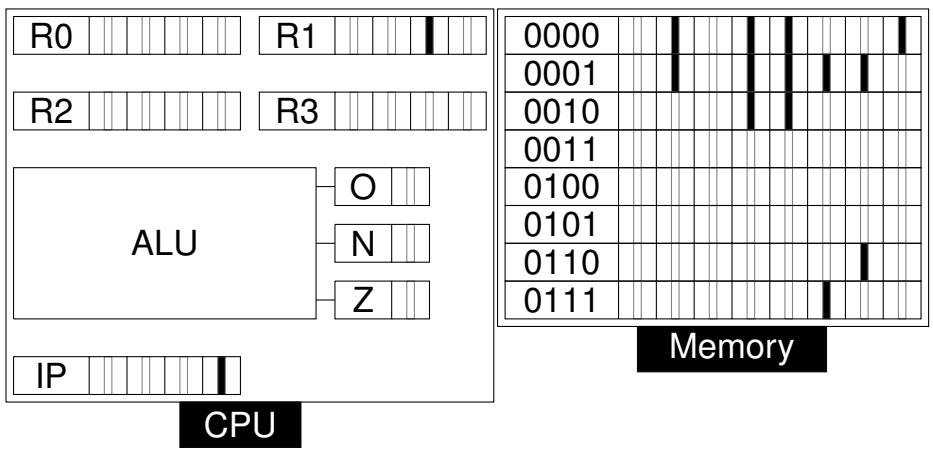
Before we run our program, let's put some numbers into our memory bank at address 0110 and 0111. Let's put the number 2 (in binary: 0010) at address 0110, and let's put the number 4 (in binary: 0100) at address 0111. When we do that, our computer now looks like this:



Notice how the register at memory address 0110 contains the number two and the register at memory address 0111 contains the number four. When our program runs, it will now have some numbers at these addresses that it can work with.

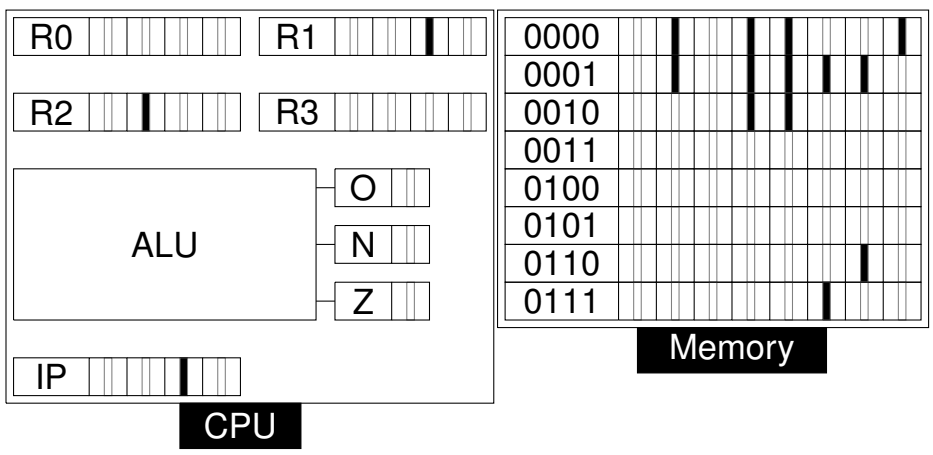
At this point, we have loaded our program (our coded instructions) into the computer's memory, and we have put some initial values into memory at the addresses 0110 and 0111, so that our program has some actual values to work with. Let's now run the program.

The instruction pointer (IP) starts at 0000, so it first looks in memory at address 0000 to find its first instruction. There it finds the instruction 01011001, which tells it to take the value stored in memory at 0110 and put it in $R1$. So, our CPU does just that: it gets the value 0001 from memory at address 0110, and it puts it into the register $R1$. Then, our CPU increments the instruction pointer (IP) to 0001. Hence, after executing the first instruction, our computer now looks like this:



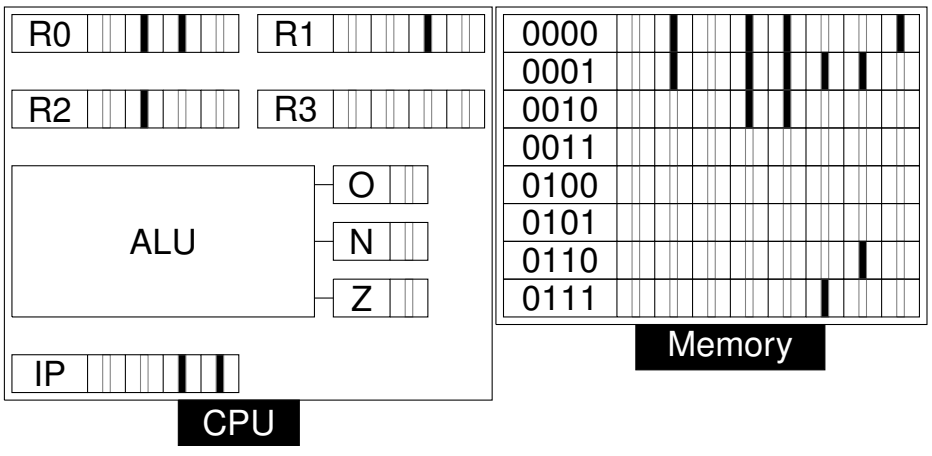
Notice how the register $R1$ contains the number two, and the instruction pointer IP now contains 0001.

Next, the CPU repeats the same process: it loads an instruction from memory, and executes it. In this case, our CPU follows the instruction pointer IP to address 0001 for the next instruction, where it sees 01011110, which tells it to take the value stored in memory at address 0111, and then put that value in register $R2$. So, our CPU does just that, then it increments the instruction pointer IP to 0010. Our computer then looks like this:



Notice that $R2$ now contains the number four (in binary 0100), and the instruction pointer IP contains 0010.

Next, our computer repeats the same procedure again: it follows the instruction pointer IP to memory address 0010, where it finds the instruction 00011000, which tells it to add the values in $R1$ and $R2$ and put the result in $R0$. So, our computer does just that: it takes the values from $R1$ and $R2$ (which are two and four respectively), and it feeds them to the ALU to add them. The ALU adds them, and it spits out the number six (in binary: 0110). No ALU flags needed to be turned on, since no overflow occurred, and the result is not negative, nor is it zero. The CPU then puts the number six into $R0$, and it increments the instruction pointer to 0011. Our computer now looks like this:



Notice that $R0$ contains our new sum: the number six 0110. At this point, there are no more instructions to execute, so our CPU has finished.

We have just used our computer like a little calculator. We put two numbers into memory (at addresses 0110 and 0111), then we ran our little program, that loads those numbers up, adds them together, and puts the result in $R0$. We could add

two other numbers by putting different numbers in memory (at addresses 0110 and 0111) to start.

58.4 Summary

IN THIS CHAPTER, we looked at how our little computer executes programs. We saw that it is pre-built to understand certain kinds of instructions, which are coded as sequences of ones and zeros.

To run a program, we essentially load the instructions into memory, and we put any initial data we might need into memory too. Then we tell the CPU to run the program, at which point it simply runs each instruction in our program, one instruction at a time, until it has finished executing each instruction in our program.

Everything that we see happening in modern computers does nothing different from what we have done here (although obviously there can be millions of lines of instructions for very complicated tasks).

Part [12]

APPENDICES

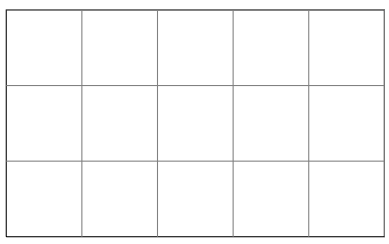
[A]

THE PYTHAGOREAN THEOREM

IN THIS APPENDIX, we will look at a famous idea in math known as the **Pythagorean Theorem**. It is a pretty neat idea. Here we will learn what it is, and why it is true (we will prove it).

A.1 The Area of Squares

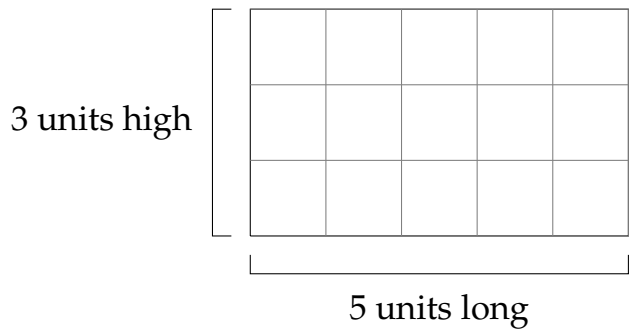
Consider the following rectangle:



How much space this rectangle takes up is called the **area** of the rectangle. How do we calculate the area of this rectangle?

Well, one way we could do it would be to count each square we see in it. If we count all the squares, we see that there are 15 squares. So this rectangle takes up 15 units of space. Its *area* is 15 of these little square units.

Another, simpler way, is to first measure its length and its height. How long is it, and how high is it? It is 5 units long, and 3 units high:



Then we multiply the two together. We can write "5 times 3" like this:

$$5 \times 3$$

We can also write it like this:

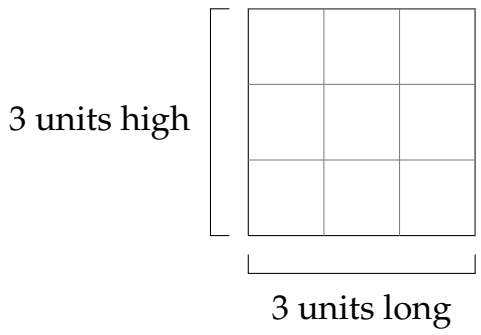
$$5 \cdot 3$$

Remark A.1. That notation should look familiar, since the "$\times$" symbol is also used for the product of sets.

Either way of writing it is fine. They both mean the same thing. Let's use the dot symbol here.

When we multiply these together, the answer is 15. And that is indeed correct. As we know from counting the square units inside the rectangle, there are exactly 15 such units in this rectangle. So, multiplying the length by the height gets us this answer too.

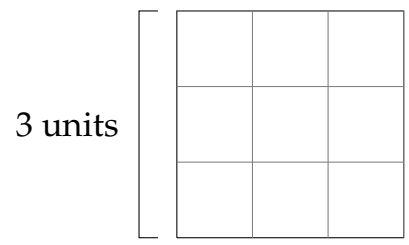
Now let's look at a square. For instance, this one:



We can calculate the area of this square in just the same way. We could either count the units inside the square individually (there are a total of 9 units), or we could multiply its length by its height, like this:

$$3 \cdot 3$$

A square is a special rectangle, in that its sides are all the very same length. So, we actually don't need to know both the length *and* the height of a square, if we want to calculate its area. All we need to know is the size of a single side. For instance:



Once we know the size of one side of the square, we can find the total area by multiplying that side by itself. In this case:

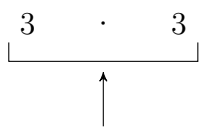
$$3 \cdot 3$$

A.2 Squaring a Length

If we are multiplying a number by itself two times, as we did with "$3 \cdot 3$," we can write it another way. We can write it like this:

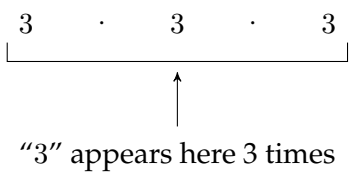
$$3^2$$

If you like, you can read that like this: "multiply the number '3' by itself 2 times." It is just another way of writing "$3 \cdot 3$" because "3" appears two times when we write that:

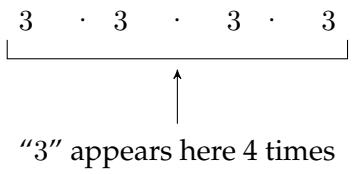


When we multiply a number by itself like this, we are essentially computing the area of a square. Hence, when we write "3^2," mathematicians say that that is the **square** of "3."

Remark A.2. If we want to multiply "3" by itself 3 times, we can write "3^3," since:



And if we want to multiply "3" by itself 4 times, we have "3^4":

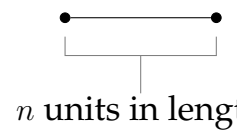


And so on, any number of times.

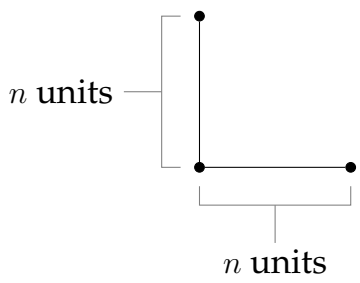
We can also use the word "square" as a verb. If you have the number "3," you can **square** it, which means that you multiply it by itself, i.e., you make it "3^2."

Think about squaring a number in a very visual way. In fact, think of it as literally taking that number and constructing a square with it.

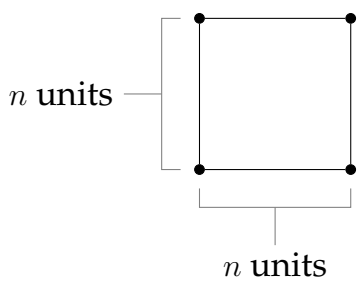
Suppose we have a number, call it n. Then suppose that we want to "square it." What do we do? First, we draw a line that is n units long:



That's the first n. Next, we draw a second n, but vertically:



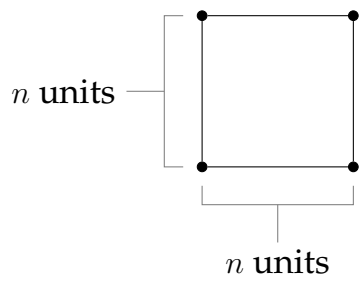
Then we make a square from these two lines:



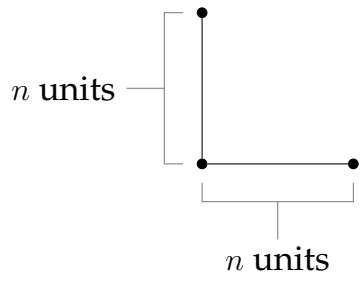
So, we start with a line of length "n," and then we "square it," i.e., we literally make a square out of it. The area of this square is "n^2," which we call the **square** of n.

A.3 Unsquaring a Square

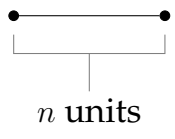
If we have a length "n" and we have squared it, we can *unsquare* it. How do we do that? Suppose we have our square of n:



To "unsquare" this is just to take it apart, and return it to the original line "n." So we remove the lines on the top right:



And then we remove one of our n-sized lines, which leaves us with a single n-sized line:



So, we end up back at n. So if "n^2" is the square, "n" is the *un*square, as it were.

To denote the "unsquare" of a square, we draw a special line over it: "$\sqrt{\ }$." Since "n^2" is a square, we can write the "unsquare" of n^2 by drawing that symbol over the top of it, like this:

$$\sqrt{n^2}$$

Mathematicians call the "unsquare" of "n^2" the **square root** of "n^2." So, the *square root* of "n^2" is just "n." It's the n-sized line we get when we take away the square we built with n:

$$\sqrt{n^2} = n$$

Example A.1. Take a line that is 5 units long. Make a square from it (i.e., 5^2). The total area of that square is 25 units. So, the square of 5 (i.e., 5^2) is 25, because $5 \cdot 5$ is 25:

$$5^2 = 25 \qquad \text{i.e.} \qquad 5 \cdot 5 = 25$$

Now, let's reverse that. We have a square with an area of 25 units. If we unsquare it (i.e., the square root of 25, $\sqrt{25}$), we get back to our original 5-unit line:

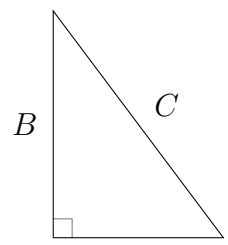
$$\sqrt{25} = 5$$

Similarly, start with a line that is 4 units long. Make a square from it (i.e., 4^2). That gives us a square with a unit of 16 units. Now unsquare it (the square root, $\sqrt{16}$). That gets us back to our original 4-unit line:

$$4 \cdot 4 = 4^2 = 16 \qquad\qquad \sqrt{16} = 4$$

A.4 The Pythagorean Theorem

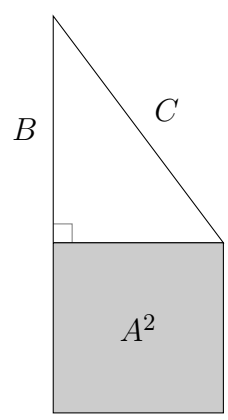
Consider the following triangle:



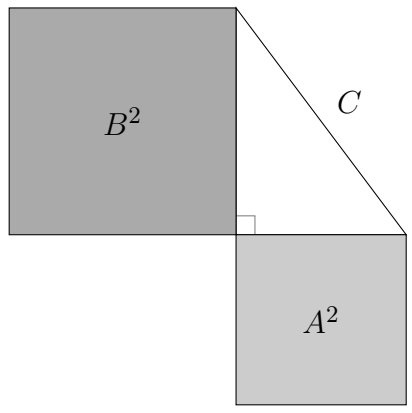
This is called a **right triangle**. It is called that because one of its three corners is a right angle. A **right angle** is just a square corner (a $90°$ angle).

The two sides A and B that join at the right angle are called the **legs** of the triangle. The side C that is opposite the right angle is called the **hypotenuse** of the triangle.

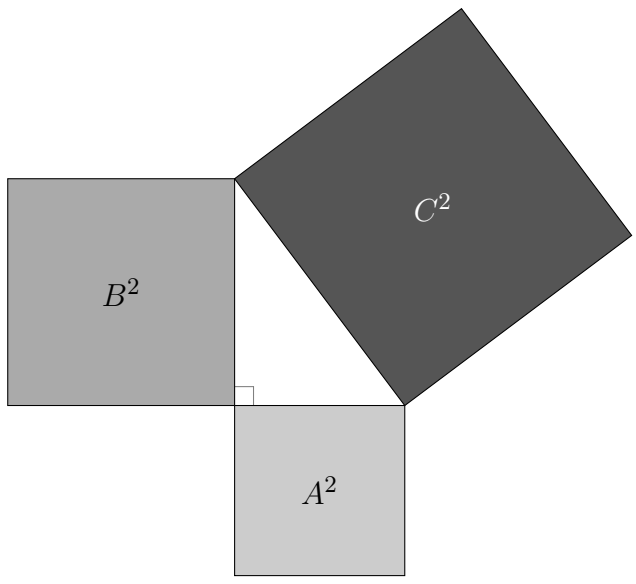
Ancient Greek mathematicians discovered that there is an interesting relationship between these sides. They discovered the following. First, notice that side A is a line segment. Let's square it, i.e., make a square from it:



Now look at side B. Let's square this length too:



Finally, look at side C. Let's square this length too:

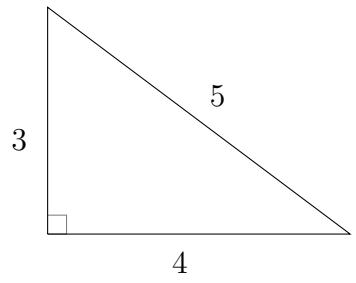


What the Greeks discovered is that if we take the area of the A-sized square and the B-sized square and we add them together, that turns out to be *exactly* the same as the area of the C-sized square. In other words:

$$A^2 + B^2 = C^2$$

This is called the **Pythogeran Theorem**.

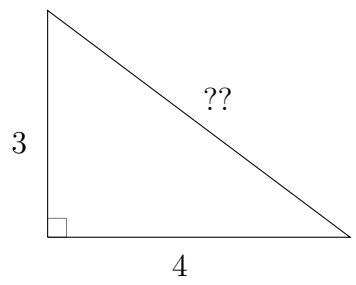
Example A.2. Suppose we have a triangle like this:



The Pythogorean Theorem tells us that if we make squares from 4, 3, and 5, and then if we add together the squares of 4 and 3, together they will have the same area as the square of 5. And indeed, that is true. Here is the calculation:

$$\begin{aligned} 4^2 + 3^2 &= 5^2 \\ 16 + 9 &= 25 \\ 25 &= 25\checkmark \end{aligned}$$

Example A.3. Suppose we have a right triangle, and we know the lengths of its legs, but we don't know the length of its hypotenuse:



Can we figure out how long the hypotenuse is? Yes, by using the Pythagorean Theorem. First, we square 4 and 3 and add them together:

$$4^2 + 3^2$$
$$16 + 9$$
$$25$$

That tells us the combined area of the square of 4 and the square of 3. The Pythagorean Theorem tells us that this will be exactly the same size as the square of the hypotenuse. Hence, we know that if we take our unknown length and square it, it will be 25:

$$??^2 = 25$$

So, we know that the area of the *square* of the hypotenuse is 25. We just want to know the length of one side of that square. We want to find the length of just the hypotenuse. To do that, we therefore need to *unsquare* 25, to get back to the original length:

$$?? = \sqrt{25}$$

And of course, the square root of 25 is 5:

$$5 = \sqrt{25}$$

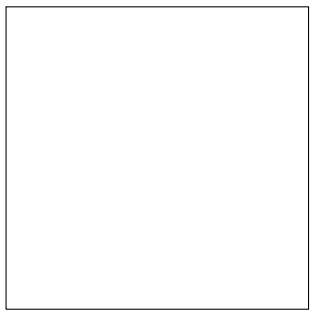
Hence, the length of the hypotenuse must be 5.

A.5 Why Is The Theorem True?

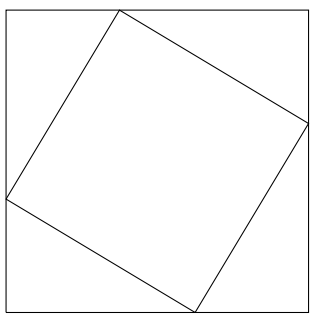
How do we know that the Pythagorean Theorem is true? There are a variety of proofs to show that it is true, but here's

a simple one.

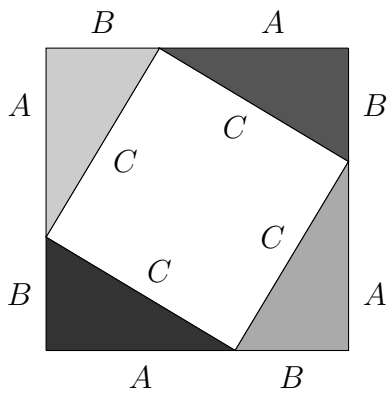
Start by drawing a square:



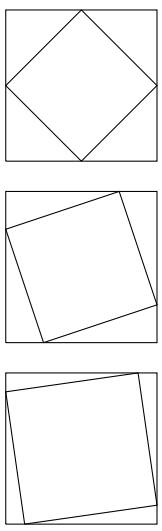
Now draw another square inside this one, but rotated a bit, like this:



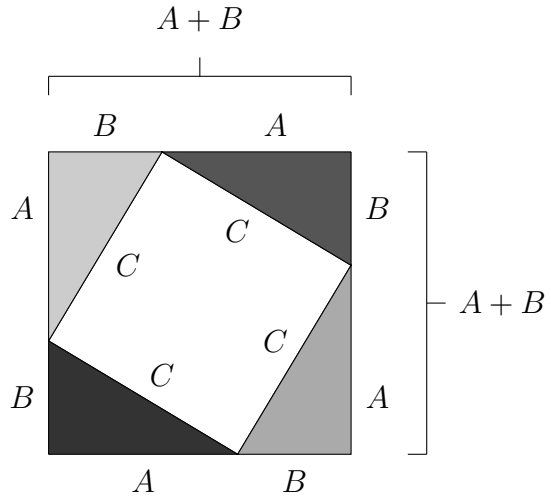
Notice that now we have a square in the middle, and four right triangles in the corners. Let's label all of the sides:



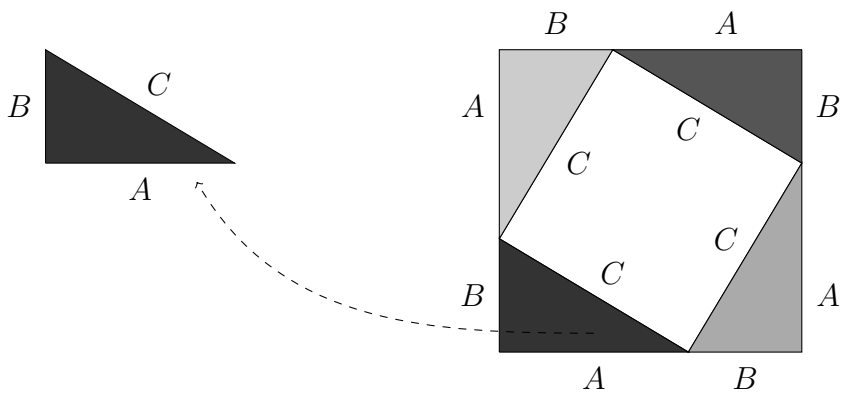
Remark A.3. There are many ways to draw the square inside the other square. For example, any of these will work fine:



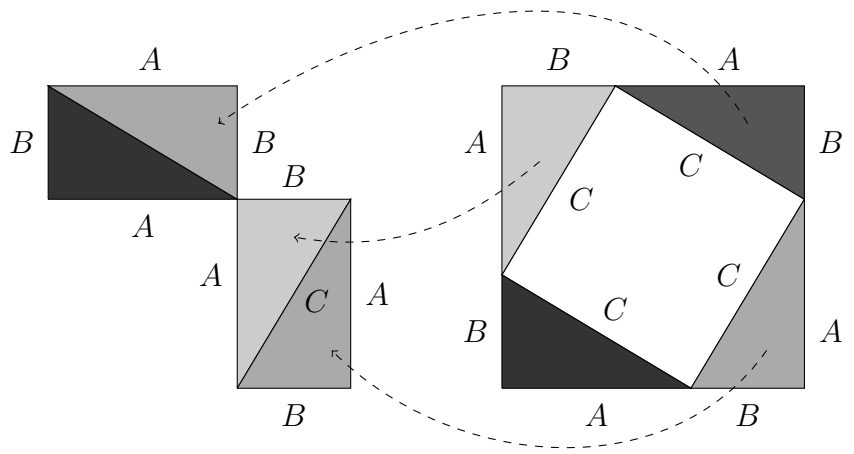
Look at each edge of the outside square. Notice that for each of the four outside edges, it has a length of $A + B$:



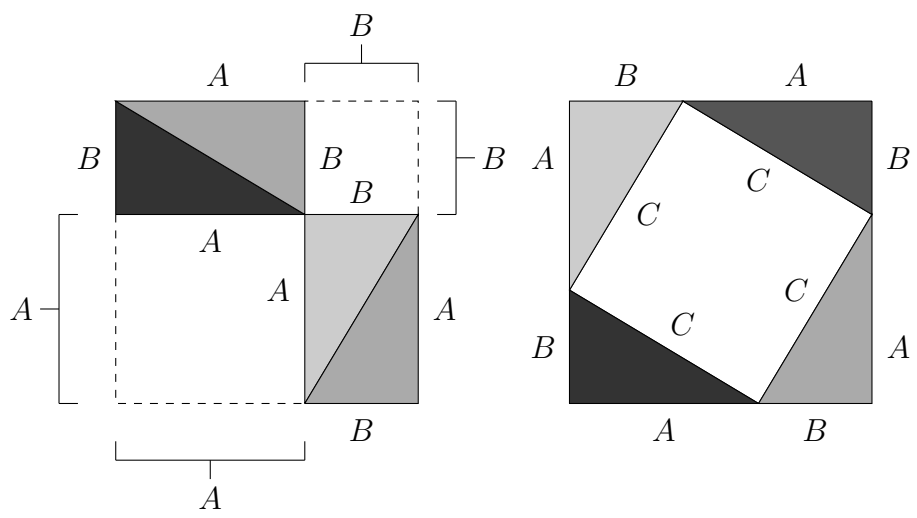
Now, let's copy the bottom left triangle over to the side:



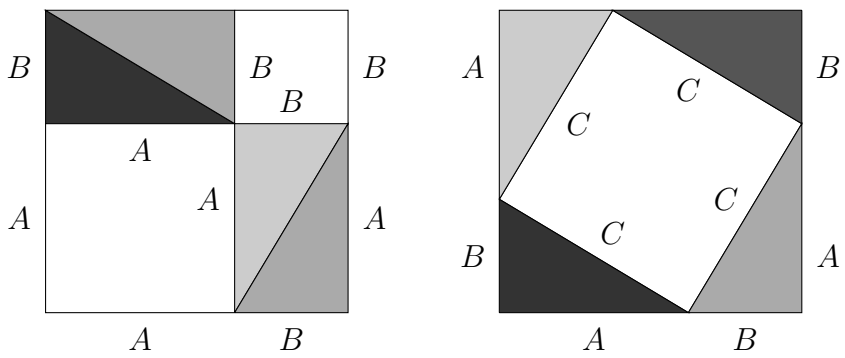
And let's copy the other triangles over to the side too, like this:



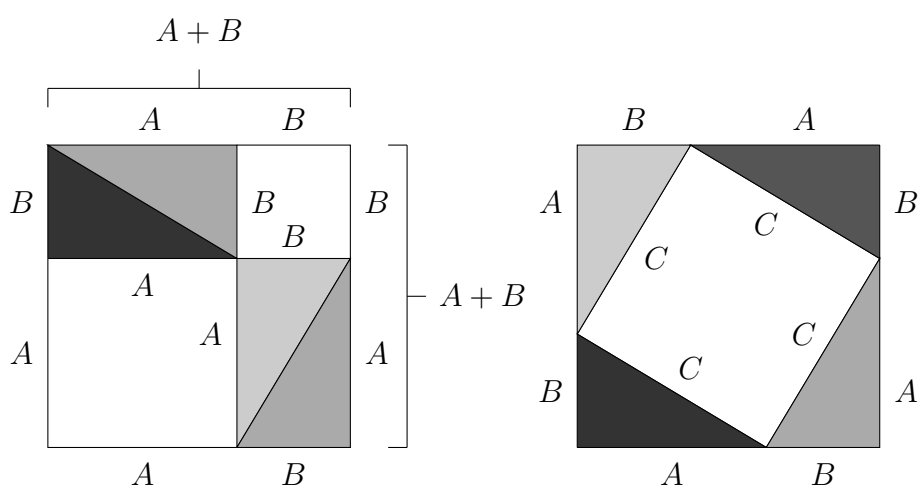
Notice that, over on the left, we have a hole for an A-sized square and another hole for a B-sized square:



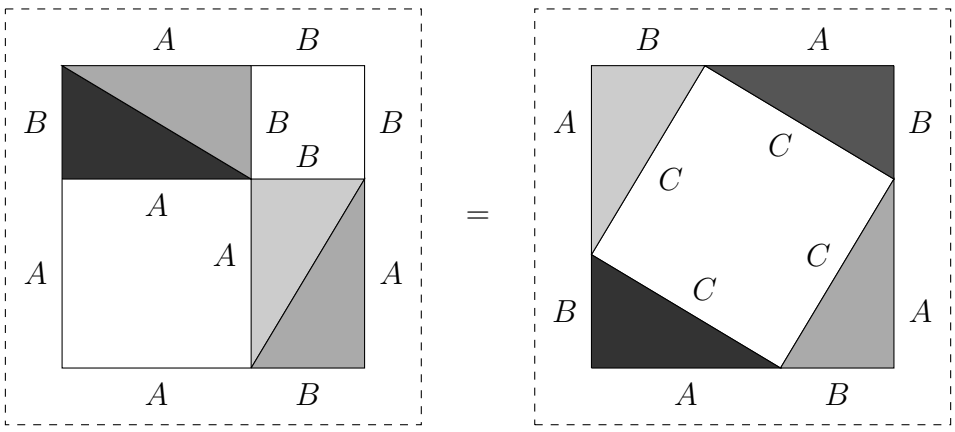
Let's fill those in:



Notice that each outside edge is made up of an A-sized line segment, and a B-sized line segment. So, the full length of each side is $A + B$:

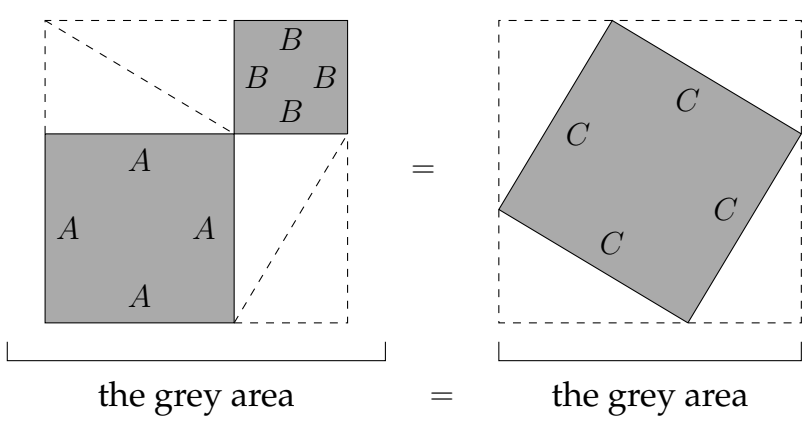


Hence, we have constructed a new square which is exactly the same size as the other one.



Remark A.4. On each side of the equals sign, there were 4 triangles, all of which were exactly the same size. By removing them, we removed exactly the same amount from each side of the equals sign in this picture. And when you remove the same amount from both sides of an equal sign, what's left are still equal. If I have two 5 pound bags of flour and I remove 2 pounds from each bag, my two bags are now smaller, but they're still equal (each of them has 3 pounds of flour left). Likewise here. If we remove four triangles from each side of the equals sign, the area that's left is smaller, but it's still equal.

Next, let's take away the $A \times B$ sized rectangles. After we remove the triangles, here's what's left, colored in with grey:



Hence, we can see that the combined area of the A-square and the B-square is the same as the area of the C-square. That is:

$$A^2 + B^2 = C^2$$

That proves the Theorem.

A.6 Summary

IN THIS CHAPTER, we looked at the **Pythagorean Theorem**. We examined what it is and what it means, and we proved it.

[B]

IRRATIONAL NUMBERS

IN THIS APPENDIX, we will look at the Greek discovery of **irrational numbers**, i.e., numbers that are not fractions.

In ancient Greece, the followers of Pythagoras allegedly believed that every number is either a whole number, or a fraction.

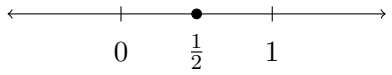
To me, this seems intuitively correct. Pick any point on the number line between 0 and 1. Surely it's some fraction. Perhaps I pick the point exactly half-way between 0 and 1. Well, that's a fraction:

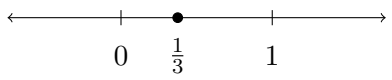
$1/2$

Or, maybe I pick a point that's exactly one-third of the way between 0 and 1. That's a fraction too:

$1/3$

Ponder. Can you think of any numbers that aren't fractions, i.e., numbers that cannot be represented as p/q?





Of course, $1/2$ and $1/3$ are nice, neat fractions. So maybe I don't pick so neatly. Maybe I pick a point at some very odd spot on the line. But surely it's a fraction too, perhaps this:

$$437/872,646,772$$

To me, it seems that any point I might pick on the line will be a fraction, even if that fraction is some very specific fraction like the one just mentioned.

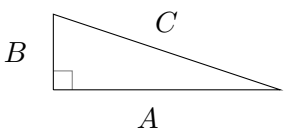
Yet, it turns out that this is not so! There are points on the number line that are *not* fractions. If we zoom way, way in on the number line, we would find that there are tiny little gaps between the fractions.

According to legend, the ancient Greeks discovered this fact, and they proved it. This must have been quite astonishing. As far as they knew at the time, the only kinds of numbers were whole numbers and fractions. But, they had proof that some numbers are neither! So what kind of number could these other numbers be?

A fraction is called a **rational number**, and numbers that are not fractions are therefore called **irrational numbers**. So, these days, we tend to say that what the Greeks discovered are *irrational numbers*.

Terminology. We now know what irrational numbers are. They are **real numbers** with no repeating pattern in their decimal expansion. As we shall see, the square root of 2 is one example, but so is π. There are lots of them. Of course, the Greeks didn't understand real numbers yet, so to them, irrational numbers must have seemed like rather mysterious entities.

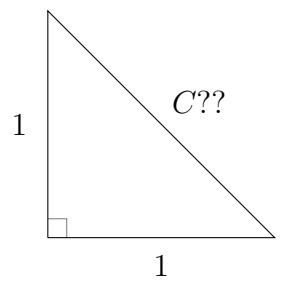
B.1 The square root of two



RECALL THE PYTHAGOREAN THEOREM, namely that for any right triangle with legs A and B and hypotenuse C, the squares of A, B, and C have a special relationship: the total area of A^2 and B^2 is the same as the total area of C^2. We write that like this:

Pythagorean Theorem : $A^2 + B^2 = C^2$

Now, suppose we are dealing with a right triangle whose legs are each exactly one unit of length. It doesn't matter what size the unit is. It could be one inch, one meter, on mile, or anything. Whatever the size of one unit, let's imagine a right triangle whose legs are each one unit long:



What is the length of C? If we use the Pythagorean Theorem, we can substitute "1" for "A" and "B":

$$1^2 + 1^2 = C^2$$

That yields this:

$$2 = C^2$$

Now we know the area of the C-square. But what we want to know is not C^2, but rather just C. We want to know the length of just one *side* of the C-square. And to figure that out, we need to "unsquare" both sides of the equation:

$$\sqrt{2} = \sqrt{C^2}$$

The square root of C^2 is just C, so:

$$\sqrt{2} = C$$

If we flip this equation around so that C is on the left side, we have this:

Remark B.1. 1^2 is $1 \cdot 1$, which is 1, so the equation reduces to this:

$$1 + 1 = C^2$$

And, of course, $1 + 1$ is 2, so we get:

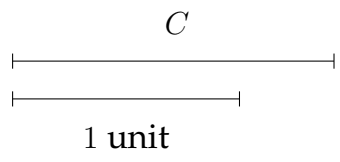
$$2 = C^2$$

Remark B.2. Recall that the square n^2 of a number n is just $n \cdot n$. The square root is the opposite of squaring. If $n \cdot n = m$, then the square root of $\sqrt{m}$ is just n. Hence, if you take the square root of a square, you just undo the square, i.e., $\sqrt{n^2} = n$. For example, $5^2 = 25$ and $\sqrt{25} = 5$, so $\sqrt{5^2} = 5$.

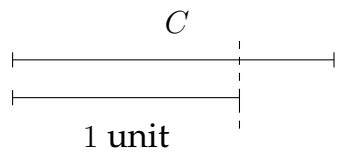
$$C = \sqrt{2}$$

Hence, the length of C is going to be the square root of 2. But what number is that, exactly?

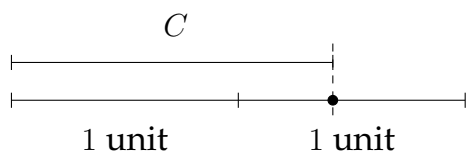
Let's take one of the legs and C, and let's lay them down side by side:



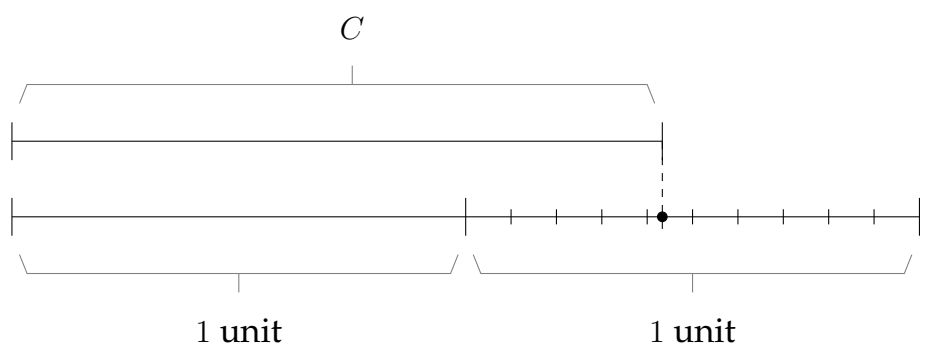
We can see that C is longer than 1:



But if we lay another 1-sized segment down, we can see that C is shorter than two of those 1-sized segments:



So, we can see that C is somewhere between 1 and 2 units. Can we calculate it exactly? Well, one thing we can do is divide up that second line segment into 10 pieces, and see where the end of C falls:



We can see that it falls somewhere between the $4/10$ mark and the $5/10$ mark. So the length of C is somewhere in between $1\ 4/10$ and $1\ 5/10$, i.e.:

$$1\ 4/10 \ < \ C \ < \ 1\ 5/10$$

Can we get more precise? Of course. We could zoom in to the stretch of line between $1\ 4/10$ and $1\ 5/10$, then we could divide it into 10 pieces again, and see where the end of C falls there. Then we could zoom in again to get more precise, and again to get more precise, and so on.

The ancient Greeks noticed that they could keep getting more and more precise like this, but they never seemed to get to an exact fraction. No matter what level of zoom they got to, the end of C always fell *in between* the tick marks.

This raises a question: will we *ever* get to a precise fraction for this number, or will it *always* fall in between the tick marks? Greek mathematicians began to suspect that maybe this particular number never would resolve to an exact fraction. It would always fall between the tick marks.

But how do you prove this? We could try and zoom in a hundred times, or a thousand times, or a million times. Suppose we do this, and suppose that each time we find that the end of C falls between tick marks. But does that prove definitively that it *always* will fall between tick marks?

No, and the reason is this: maybe we just haven't zoomed in enough. Maybe we need to zoom in a gazillion times before we finally get to that tick mark. If we want to be absolutely sure that it never falls on a tick mark, we need to *prove* that it's *impossible* to ever get to tick mark.

That is just what the ancient Greeks did. They proved that the square root of 2 cannot ever fall on a tick mark. And that is surely astonishing. This means that some numbers (like this one, the square root of 2) fall "between" the fractions! It means that there are *more* numbers beyond the fractions, that the Greeks didn't know about, which fill in all the gaps!

B.2 Some preliminaries

WE WILL PROVE that there are such "irrational" numbers. But first, let us establish some preliminaries. To begin, let us accept the following facts as axioms. We could prove these, but we will just take them as given for our purposes here.

Fact B.1. If we know that some quantity n is equal to some other quantity m, then we can substitute m in the place of n wherever we encounter n. For instance, if we know that $n = 5 \cdot x$, and we later come across $n+6$, then we can substitute $5 \cdot x$ in for n, to get $(5 \cdot x) + 6$.

Fact B.2. If an integer n is even, then it is twice some other integer k. Hence, if we have an even integer n, we can rewrite it as $2 \cdot k$. For instance, 10 is even, and it is twice the amount of 5. So instead of writing 10, we can write $2 \cdot 5$. 26 is even, and it is twice another integer, namely 13. Instead of writing 26, we can write $2 \cdot 13$.

Fact B.3. If an integer n is odd, then it is twice some other integer k, plus 1. Hence, if we have an odd integer n, we can

rewrite it as $(2 \cdot k)+1$. For instance, 13 is odd, and it is twice the amount of 6, plus 1. So, instead of writing 13, we can write $(2 \cdot 6) + 1$.

Fact B.4. Every integer n is either even or odd. It can't be both. It must be one or the other.

Fact B.5. Adding two integers n and m yields another integer. For example, if we add 5 and 3 (both integers), we get 8 (another integer). Hence, if we see $n + m$, and we know that n and m are integers, then we know that the result of $n + m$ is also an integer.

Fact B.6. Multiplying two integers n and m yields an integer, just as addition does.

Fact B.7. We are allowed to do basic algebraic arithmetic with expressions containing variables. For example, if we have $2 \cdot (2 \cdot x)$, we can evaluate that to $4 \cdot x$, or if we have $(3 \cdot x) + (2 \cdot x)$, we can reduce that to $5 \cdot x$.

Fact B.8. We are allowed to manipulate equations, so long as we do the same thing to both sides. We can add the same number to both sides of an equation, we can subtract the same number to both sides of an equation, and so on.

Fact B.9. We can factor out a 2 from various equations. For instance, if we have $6 \cdot x$, we can factor out 2 to get $2 \cdot (3 \cdot x)$. If we have $(2 \cdot x) + (2 \cdot y)$, we can factor out 2 to get $2 \cdot (x + y)$. If we have $(4 \cdot x) + (2 \cdot y)$, we can factor out 2 to get $2 \cdot ((2 \cdot x) + y)$.

Fact B.10. If we have $(a + b) \cdot (c + d)$, we can rewrite that like this: $(a \cdot c) + (a \cdot d) + (b \cdot c) + (b \cdot d)$.

B.3 Some lemmas

LET'S NOW PROVE a few lemmas that will be useful when we look at the square root of 2. The first lemma is this: if an integer n is even, then its square n^2 is even too. Let's write this down as a lemma:

Lemma B.1. For any integer n, if n is even, then n^2 is even.

Proof. How do we prove this? We use the facts given above, and logic. So, let us assume that we have some arbitrarily chosen integer n, which is even. We don't care which even n it is. We pick it blind-folded, so to speak. All we know is that it is even. What we want to prove is that n^2 is also even.

What do we do next? Well, if n is even, then Fact B.2 tells us that n is therefore twice some other integer k. We don't really care what k is. All we need to know right now is that n is twice some other integer k. So, let's rewrite n as twice the amount of k:

(by Fact B.2) $$n = 2 \cdot k$$

Since n is the same as $2 \cdot k$, Fact B.1 tells us that we can substitute $2 \cdot k$ for n, anytime we see n. Well, we want to prove that n^2 is even, and we can see there's an n in there. So, let's substitute "$2 \cdot k$" for "n" in "n^2":

(by Fact B.1) $$n^2 = (2 \cdot k)^2$$

Fact B.7 says we can use basic algebraic arithmetic. What is $(2 \cdot k)^2$? Anything squared is just itself multiplied by itself. So, $(2 \cdot k)^2$ is the same as $(2 \cdot k) \cdot (2 \cdot k)$. Hence:

(by Fact B.7) $$n^2 = (2 \cdot k) \cdot (2 \cdot k)$$

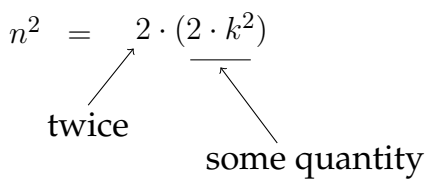
We can do some more basic algebraic arithmetic. What is $(2 \cdot k) \cdot (2 \cdot k)$? It's $4 \cdot k^2$:

(by Fact B.7) $$n^2 = 4 \cdot k^2$$

We know from Fact B.9 that we can factor 2 out from $4 \cdot k^2$, which gives us $2 \cdot (2 \cdot k^2)$:

(by Fact B.9) $$n^2 = 2 \cdot (2 \cdot k^2)$$

At this point, look at the shape of the right hand side of the equation. We can see here that we have twice some quantity. We have *twice* the amount of whatever quantity "$(2 \cdot k^2)$" evaluates to:



According to Fact B.2, if a quantity is 2 times some integer, then it's even. Well, if "$(2 \cdot k^2)$" evaluates to an integer, then that means that n^2 will evaluate to two times an integer, which means n^2 will evaluate to an even number. So, is "$(2 \cdot k^2)$" an integer?

Yes, it is. We know that k is an integer, because we said above that since n is even, it must be twice some other integer k (by Fact B.2). Further, what's k^2? It's really just $k \cdot k$, and we know from Fact B.6 that multiplying one integer by another integer gives us another integer. So, $k \cdot k$ (i.e., k^2) is also an integer. And, multiplying 2 (an integer) by k^2 (an integer), gives us an integer too, again because of Fact B.6. So "$(2 \cdot k^2)$" must be an integer.

Hence, with "$2 \cdot (2 \cdot k^2)$," we do indeed have twice an integer. That is to say, we have an expression with the shape of "$2 \cdot k$," it's just that "k" is "$(2 \cdot k^2)$." So by Fact B.2, we know that this expression therefore represents some even integer:

(by Fact B.2) $\qquad 2 \cdot (2 \cdot k^2)$ is even

Thus, since n^2 is equal to "$2 \cdot (2 \cdot k^2)$," we can conclude that n^2 must be even:

n^2 is even

With that, we have finished our proof. We have shown that if n is even, then n^2 must also be even.

□

Next, let's prove that if an integer n is *odd*, then its square n^2 will be odd too. Let's put that down as a lemma:

Lemma B.2. For any integer n, if n is odd, then n^2 is odd.

Proof. We can prove this lemma much like we did the last one. It's just that in this case, n is odd, so we don't substitute in $2 \cdot k$. Rather, we substitute in $(2 \cdot k) + 1$.

So, as before, we begin by picking some arbitrary integer n, which is odd. Again, we don't care which odd integer n is. We pick it blind-folded. All we know is that it is odd.

Fact B.3 tells us that if n is odd, then it must be twice some other integer k plus one. Hence, we know that n can be rewritten as $(2 \cdot k) + 1$:

(by Fact B.3) $\qquad n = (2 \cdot k) + 1$

Since we know that n is the same as $(2 \cdot k) + 1$, Fact B.1 tells us that we can subsitute $(2 \cdot k) + 1$ for n, anytime we see n. We

want to prove that n^2 is odd, and we can see an n in there. So, let's take n^2, and substitute "$(2 \cdot k) + 1$" in for "n":

(by Fact B.1) $$n^2 = ((2 \cdot k) + 1)^2$$

What's $(2k+1)^2$? Fact B.7 tells us we can do basic algebraic arithmetic on this expression, and anything squared is just itself multiplied by itself. Hence, "$((2 \cdot k) + 1)^2$" must be "$(2 \cdot k) + 1$" multiplied by itself:

(by Fact B.7) $$n^2 = ((2 \cdot k) + 1) \cdot ((2 \cdot k) + 1)$$

What do we do next? We want to do some more algebraic manipulations and calculations. We can see that the right hand side of this expression has the shape of "$(a + b) \cdot (c + d)$," it's just that "a" is "$(2 \cdot k)$," "b" is "1," "c" is "$(2 \cdot k)$," and "d" is "1." Well, Fact B.10 says we can rewrite the right hand side of our equation so that it has the shape "$(a \cdot c) + (a \cdot d) + (b \cdot c) + (b \cdot d)$," like this:

(by Fact B.10)
$$\begin{aligned} n^2 = &((2 \cdot k) \cdot (2 \cdot k)) \\ &+ ((2 \cdot k) \cdot 1) \\ &+ (1 \cdot (2 \cdot k)) \\ &+ (1 \cdot 1) \end{aligned}$$

Then, using basic algebraic arithmetic (Fact B.7), we can reduce "$((2 \cdot k) \cdot (2 \cdot k))$" to "$4 \cdot k^2$," and we can reduce each of "$((2 \cdot k) \cdot 1)$" and "$(1 \cdot (2 \cdot k))$ to "$2 \cdot k$." Like this:

(by Fact B.7) $$n^2 = (4 \cdot k^2) + (2 \cdot k) + (2 \cdot k) + 1$$

We can again use basic algebraic arithmetic (Fact B.7) to reduce "$(2 \cdot k) + (2 \cdot k)$" to "$(4 \cdot k)$":

(by Fact B.7) $$n^2 = (4 \cdot k^2) + (4 \cdot k) + 1$$

Next, Fact B.9 tells us that we are allowed to factor out 2 whenever we have the opportunity, and here we can factor a 2 out of "$(4 \cdot k^2) + (4 \cdot k)$" to get $2 \cdot ((2 \cdot k^2) + (2 \cdot k))$:

(by Fact B.9) $$n^2 = (2 \cdot ((2 \cdot k^2) + (2 \cdot k))) + 1$$

At this point, the right hand side of the equation looks like it has the shape of "$(2 \cdot k) + 1$" except that instead of "k" we have "$((2 \cdot k^2) + (2 \cdot k))$." Fact B.3 tells us that if an expression has the shape "$(2 \cdot k) + 1$" for some integer k, then it evaluates to an odd number. So, if "$((2 \cdot k^2) + (2 \cdot k))$" evaluates to an integer, then the whole expression "$(2 \cdot ((2 \cdot k^2) + (2 \cdot k))) + 1$" will evaluate to an *odd* integer.

So, does "$((2 \cdot k^2) + (2 \cdot k))$" evaluate to an integer? The answer is yes. We know that k is an integer, because we said above that since n is odd, it must be twice some other integer k plus one (by Fact B.3). We also know that k^2 must be an integer, because k^2 is just $k \cdot k$, and multiplying an integer by an integer yields another integer (by Fact B.6). We also know that $2 \cdot k^2$ must be an integer, because 2 is an integer and k^2 is an integer, and so multiplying the one by the other must yield an integer (by Fact B.6 again). We know that $2 \cdot k$ must be an integer as well, because 2 is an integer and k is an integer, and multiplying them together yields an integer. Finally, we know that $((2 \cdot k^2) + (2 \cdot k))$ must be an integer, because Fact B.5 tells us that adding an integer ($2 \cdot k^2$) to another integer ($2 \cdot k$) yields an integer. Hence, the whole expression $((2 \cdot k^2) + (2 \cdot k))$ must evaluate to an integer.

So, with "$(2 \cdot ((2 \cdot k^2) + (2 \cdot k))) + 1$," we do indeed have something that evaluates to twice an integer plus one. And that means that it must be odd:

(by Fact B.3) $(2 \cdot ((2 \cdot k^2) + (2 \cdot k))) + 1$ is odd

Thus, since n^2 is equal to "$(2 \cdot ((2 \cdot k^2) + (2 \cdot k))) + 1$," we can conclude that n^2 must therefore be odd:

n^2 is odd

With that, we have finished our proof. We have shown that if n is odd, then n^2 must also be odd.

□

Let's prove one more lemma. Let's prove that if an integer n^2 is even, then n must be even too. We have already proven that if n is even, then its square must be even. But we haven't proved the other direction. We haven't proven that if n^2 is even, then n must be even. Let's prove that now. First, let's state the lemma:

Lemma B.3. For any integer n, if n^2 is even, then n is even.

Proof. It's easy for us to express the idea of this proof in somewhat casual English prose. We can say: "well, if n^2 is even, then n must be even, because if n were odd, then n^2 would be odd." But let's make the steps of such reasoning more explicit.

This is a proof by contradiction. So first, let's assume the opposite of what we want to prove. We want to prove that whenever n^2 is even, n must be even too. The opposite of that is a case where n^2 is even, yet n is odd. So let's assume that this is so:

(Assumption) n^2 is even and n is odd

Next, we need to show that this is an impossible scenario. We want to show that it is impossible for n^2 to be even, and

yet n to be odd. How do we do that? We derive a contradiction.

Notice that we assumed n is odd. However, we know from Lemma B.2 that if n is odd, then n^2 must be odd too. Hence:

(by Lemma B.2) n^2 is odd

But we originally assumed that n^2 is even:

(from our Assumption) n^2 is even

Fact B.4 tells us that every integer must be either even, or odd. Hence, this is a contradiction. n^2 can't be both even and odd. It must be one or the other.

Thus, it is impossible for n^2 to be even, and for n to also be odd. Therefore, our original assumption must be wrong. If n^2 is even, it must be the case that n is even too.

□

B.4 $\sqrt{2}$ is irrational

WE ARE NOW READY to prove that the square root of 2 is irrational (i.e., that it is not a fraction). Let's put this down as a theorem:

Theorem B.1. $\sqrt{2}$ is irrational.

Proof. To prove this, let's do a proof by contradiction. That is to say, let's assume the opposite of what we want to prove, and show that it's impossible.

We're trying to prove that $\sqrt{2}$ is irrational, so the opposite of that is this: $\sqrt{2}$ *is* rational. So, let's assume precisely that: let's assume that $\sqrt{2}$ is a fully reduced fraction. If it's a fully

reduced fraction, that means we can write it as p/q, where p and q are integers, and they have no common divisor that we might use to reduce the fraction further:

(Assumption) $\sqrt{2} = \frac{p}{q}$, a fully reduced fraction

Next, let's get rid of the square root sign on the left. Fact B.8 tells us we can manipulate equations, so long as we do the same thing to both sides of the equation. In this case, let's square both sides of the equation:

(by Fact B.8) $$(\sqrt{2})^2 = (\frac{p}{q})^2$$

Using basic algebraic arithmetic (Fact B.7), we can simplify this further. Squaring a square root just removes the square root, and squaring a fraction squares the top and bottom of the fraction. Like this:

(by Fact B.7) $$2 = \frac{p^2}{q^2}$$

Next, we can multiply both sides of the equation by q^2, which is allowed by Fact B.8 because we are doing the same thing to both sides of the equation:

(by Fact B.8) $$2 \cdot q^2 = \frac{p^2}{q^2} \cdot q^2$$

We can now use basic algebraic arithmetic (Fact B.7). On the right side, q^2 gets canceled out, so that we are left with:

(by Fact B.7) $$2 \cdot q^2 = p^2$$

Let's flip the equation around, so that p^2 is on the left hand side:

(by Fact B.7) $p^2 = 2 \cdot q^2$

Notice now that, on the right hand side, we have 2 times q^2. In other words, p^2 is twice some quantity. This has the shape of an even number $2 \cdot k$, it's just that instead of k, we have q^2. Hence, if q^2 evaluates to an integer, then the right hand side of the equation will evaluate to twice an integer, and we can conclude that p^2 must therefore be even.

So, does q^2 evaluate to an integer? It does! We know that q is in an integer, because we started by assuming that p and q are integers. Also, q^2 is just $q \cdot q$, and Fact B.6 tells us that multiplying an integer by an integer yields an integer. So, q^2 evaluates to an integer. Hence, p^2 is twice some integer, and hence p^2 must be even:

(by Fact B.2) p^2 is even

We know from Lemma B.3 that if p^2 is even, then p must be even too. Hence, we can write p as twice some integer r:

(by Lemma B.3 and Fact B.2) $p = 2 \cdot r$

Since p is the same as $2 \cdot r$, Fact B.1 tells us that we can substitute $2 \cdot r$ for p, anytime we see p. Before, we had $2 \cdot q^2 = p^2$, and we can see a "p" in there. Let's substitute $2 \cdot r$ in for p:

(by Fact B.1) $2 \cdot q^2 = (2 \cdot r)^2$

Next, we can do some basic algebraic arithmetic (Fact B.7). What's $(2 \cdot r)^2$? We can reduce it to $4 \cdot r^2$:

(by Fact B.7) $$2 \cdot q^2 = 4 \cdot r^2$$

Fact B.8 tells us we can do whatever we want to both sides of an equation, so long as we do the same thing to both sides. So, let's divide both sides of this equation by 2:

(by Fact B.8) $$\frac{2 \cdot q^2}{2} = \frac{4 \cdot r^2}{2}$$

Let's use basic algebraic arithmetic to simplify this (Fact B.7). On the left side, 2 gets canceled out, and on the right side, we end up with $2 \cdot r^2$:

(by Fact B.7) $$q^2 = 2 \cdot r^2$$

Notice now that, on the right hand side of this equation, we have 2 times $r \cdot^2$. In other words, we have an expression that has the shape $2 \cdot k$ again, where k is r^2. So, if r^2 evaluates to an integer, then q^2 will evaluate to twice an integer, and therefore q^2 will be even.

So, does r^2 evaluate to an integer? Again, the answer is yes. We know already that r is an integer, because we deduced that p is even, and that means it is twice some integer r. Also, r^2 is just $r \cdot r$, and Fact B.6 tells us that multiplying an integer by an integer yields an integer. So, r^2 evaluates to an integer.

Hence, q^2 evaluates to twice some integer, and according to Fact B.2, that means q^2 must be even:

(by Fact B.2) $$q^2 \text{ is even}$$

We also know from Lemma B.3 that if q^2 is even, then q must be even too. And according to Fact B.2, if q is even, then that means it is twice some integer t:

(by Lemma B.3 and Fact B.2) $q = 2 \cdot t$

Since we know that $p = 2 \cdot r$ and we know that $q = 2 \cdot t$, Fact B.1 tells us that we can substitute "$2 \cdot r$" and "$2 \cdot t$" for "p" and "q" (respectively), anytime we see "p" and "q." At the beginning, we started by assuming that $\sqrt{2}$ is a fully reduced fraction, p/q, and we can see a "p" and a "q" in there. Let's substitute "$2 \cdot r$" and "$2 \cdot t$":

(by Fact B.1) $$\sqrt{2} = \frac{2r}{2t}$$

What can we say about the fraction on the right hand side of the equation? Notice that it is *not* fully reduced. Why? Because both the top and bottom number of this fraction are even numbers, and so they could be divided by 2 to reduce the fraction.

But that contradicts our original assumption. We started by assuming that $\sqrt{2}$ is a fully reduced fraction, and that contradicts what we just deduced: that $\sqrt{2}$ is not a fully reduced fraction.

So, we've ended up in an impossible state of affairs. This scenario is impossible. It cannot be that $\sqrt{2}$ is a fraction.

□

That completes our proof. We proved that the square root of two is not a fraction. We did it by assuming that it is a fully reduced fraction, and then we showed that this is impossible. Hence, the square root of two simply cannot be a fraction.

B.5 Summary

IN THIS APPENDIX, we looked at the ancient Greek discovery of irrational numbers. The Pythagorean theorem tells us that if we have a right triangle with legs that are one unit long, then the hypotenuse should have a length that is the square root of two. But the square root of two does not look to be a fraction, and we proved that in fact it is impossible for it to be a fraction. Hence, the square root of two is a number that falls "between" fractions.

[C]

ALGEBRA TRICKS

PERHAPS YOU REMEMBER from algebra classes in school that you can do the same thing to both sides of an equation, or that you can cancel certain things out. In this appendix, we will look at some of these basic computing tricks that we can do with algebra, and we will examine why it is that we can do them.

C.1 There Is Only One Identity

A GROUP OR A MONOID has an identity. But can a group or monoid have more than one identity? The answer is no. Every group or monoid has exactly one identity, no more, no less. We can prove this fact, with a proof by contradiction.

We want to show that there is only one identity, and so we want to start our proof by assuming the opposite of that. Hence, let's start by supposing that there are two distinct identities. Let's call them e_1 and e_2.

Recall that an identity is a "do-nothing" element. You can combine it with any other element, and you'll get that other element back:

$$\underbrace{x}_{\text{any element } x} \star \underbrace{e_1}_{\text{a "do-nothing" element}} = \underbrace{x}_{\text{that same element } x}$$

It works the other way around too:

$$\underbrace{e_1}_{\text{a "do-nothing" identity}} \star \underbrace{x}_{\text{any element } x} = \underbrace{x}_{\text{that same element } x}$$

Well, let's put e_2 in place of x. Since e_1 is a "do-nothing" element, that means it will do nothing, and we will get back e_2:

$$\underbrace{e_1}_{\text{a "do-nothing" identity}} \star \underbrace{e_2}_{\text{the element } e_2} = \underbrace{e_2}_{\text{that same element } e_2}$$

So we know that $e_1 \star e_2 = e_2$. File that little fact away for the time being. We'll return to it in a moment.

Next, recall that we supposed at the beginning of our proof that e_2 is an identity too, so that means that it is also a "do-nothing" element. Hence, if we combine e_2 with any element x, it will do nothing, and we'll get back x:

$$\underbrace{x}_{\text{any element } x} \star \underbrace{e_2}_{\text{a "do-nothing" element}} = \underbrace{x}_{\text{that same element } x}$$

Let's put e_1 in place of x. Since e_2 is a "do-nothing" element, that means it will do nothing, and we will get back e_1:

$$\underbrace{e_1}_{\text{the element } e_1} \star \underbrace{e_2}_{\text{a "do-nothing" element}} = \underbrace{e_1}_{\text{that same element } e_1}$$

So we know that $e_1 \star e_2 = e_1$. But wait, we also deduced a moment ago that $e_1 \star e_2 = e_2$. So, on our current assumptions, $e_1 \star e_2$ evaluates to e_1, but it *also* evaluates to e_2.

What does this tell us? We know that "$\star$" is a **function**, and that means that it will always map "$e_1 \star e_2$" to one and the same result. Hence, since it maps "$e_1 \star e_2$" to both e_1 and e_2, that means e_1 and e_2 must be one and the same element.

Hence, e_1 and e_2 cannot be two distinct elements after all, which contradicts our original assumption that they are *distinct* identities. So, we have shown that it is impossible for there to be two distinct identities. Therefore, we can conclude that there is only one identity in a group or monoid.

C.2 Do the Same on Both Sides

YOU MAY REMEMBER from algebra classes in school that we can always do the same thing to both sides of an equation. Let's talk about why this is.

Suppose we have an algebra defined by the following Cayley table:

$\star$	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

We can see from this table that if we combine "b" and "c" we get "a":

$$b \star c = a$$

We can also see that if we combine "c" and "a" we also get "a":

$$c \star a = a$$

So, "$b \star c$" and "$c \star a$" evaluate to the same element, namely "a." Hence, we can say that these two operations produce *equal* results, which we write like this:

$$b \star c \quad = \quad c \star a$$

Think about what this equation asserts. It is asserting that both sides of the equation refer to the same element:

$$\underbrace{b \star c}_{\text{the element } a} \quad = \quad \underbrace{c \star a}_{\text{the same element } a}$$

Now, suppose that I want to apply (say) a "b" to both sides of this equation. Like this:

$$b \quad \star \quad \underbrace{(b \star c)}_{\text{the element } a} \quad = \quad b \quad \star \quad \underbrace{(c \star a)}_{\text{the same element } a}$$

Can I do this? Is this a legitimate move? Yes, it is, but let's think about why.

What I did, was I added a "b" to both sides of the equation. Hence, on the left hand side of the equation, I am now combining "b" with the element "a," and on the right hand side of the equation, I am now combining "b" with the same element "a":

$$\underbrace{b}_{\text{the element } b} \quad \star \quad \underbrace{(b \star c)}_{\text{the element } a} \quad = \quad \underbrace{b}_{\text{the element } b} \quad \star \quad \underbrace{(c \star a)}_{\text{the same element } a}$$

Are these two sides equal? Of course! Combining "b" with "a" on the left hand side will give me the same value as combining "b" with "a" on the right hand side. If we look in the Caley table, we will see that both sides will evaluate to "b."

Why does it turn out to be equal like this? Because "$\star$" is a **binary operation**, which by definition is a **function**. Because

it is a function, we know that it will always map the same pair of elements to the same result.

It doesn't really matter what we have in place of "$(b \star c)$" and "$(c \star a)$." Whatever they evaluate to, so long as they evaluate to the same value — it doesn't matter which element it is, so let's just call it x — then we're just applying "b" to that same element x:

$$\underbrace{b}_{\text{the element } b} \star \underbrace{(b \star c)}_{\text{some element } x} = \underbrace{b}_{\text{the element } b} \star \underbrace{(c \star a)}_{\text{the same element } x}$$

And "$b \star x$" will always equal "$b \star x$," no matter which element x from our carrier set x happens to be. This is why we can apply "b" to both sides, and the equation is still a true assertion.

For the same reason, we can apply "a" or "c" to both sides of the equation. It needn't be "b." It doesn't matter if we apply "a," "b," or "c" to both sides of the equation — and since it doesn't matter which one, let's just call it "y":

$$\underbrace{c}_{\text{some element } y} \star \underbrace{(b \star c)}_{\text{some element } x} = \underbrace{c}_{\text{the same element } y} \star \underbrace{(c \star a)}_{\text{the same element } x}$$

If "$(b \star c)$" and "$(c \star a)$" are equal already, then as long as we apply the same value y to both sides of the equation, the fact that "$\star$" is a function means that it will always give us the same result, so the equation will still be true.

C.3 Cancellation

IN THE LAST SECTION we showed why we can apply the same element to both sides of an equation. But we can also cancel out the same thing from both sides of an equation. Suppose we have an equation that looks like this (the names of the elements don't matter, so we'll just use x, y, and z):

$$x \star y = x \star z$$

We can cancel out the "x" from both sides:

$$\cancel{x} \star y = \cancel{x} \star z$$

And that leaves us with this:

$$y = z$$

We call this the **cancellation law** for groups. Let's write the whole idea down more concisely. Let's say that for any x, y, z in our carrier set:

$$\text{if} \quad x \star y = x \star z \quad \text{then} \quad y = z$$

Let's show *why* this is true for any group, by proving it. Let's start with what we are given. Let's start by assuming that $x \star y$ is the same as $x \star z$:

$$\underbrace{x \star y}_{\text{this element}} \quad \underbrace{=}_{\text{is the same as}} \quad \underbrace{x \star z}_{\text{this element}}$$

What we want to show is that, whenever this is the case, then it must be the case that y is equal to z too.

First, let's apply the inverse of x to both sides. We can do this because we can apply anything we like to both sides of the equation, so long as we apply the same thing to both sides:

$$\underbrace{x^{-1}}_{\text{apply the inverse of } x} \star (x \star y) = \underbrace{x^{-1}}_{\text{apply the inverse of } x} \star (x \star z)$$

Next, let's re-arrange the parentheses. We can do this because we know that groups are associative, which means where the parentheses go doesn't matter:

$$\underbrace{(x^{-1} \star x)}_{\text{regroup the parentheses}} \star y = \underbrace{(x^{-1} \star x)}_{\text{regroup the parentheses}} \star z$$

Next, notice that we are now combining an element x and its inverse x^{-1} on both sides of this equation. And any element combined with its inverse evaluates to the identity element.

$$\underbrace{(x^{-1} \star x)}_{\text{evaluates to the identity}} \star y = \underbrace{(x^{-1} \star x)}_{\text{evaluates to the identity}} \star z$$

So we can rewrite this. It doesn't matter what the identity element is here, so let's just call it e. Hence, we have:

$$e \star y = e \star z$$

Next, notice that now we are combining y with the identity element (on the left hand side of the equation), and we are combining z with the identity element (on the right hand side of the equation). Recall that the identity element is a "do-nothing" element. Combining the identity element with any other element just gives us back that other element:

$$\underbrace{e \star y}_{\text{this just evaluates to } y} = \underbrace{e \star z}_{\text{this just evaluates to } z}$$

So, we can rewrite our equation without the "e," to get this:

$$y = z$$

Hence, in the end, y must be equal to z. And that was exactly what we wanted to prove, so we have completed our proof.

We have shown that, if $x \star y$ is equal to $x \star z$, then y must be equal to z as well, and hence we can just ignore or cancel out the "x" on both sides of the equation.

We can use exactly the same technique to prove that we can cancel "x" if it appears on the right side too. That is, we can prove this:

$$\text{if} \quad y \star x = z \star x \quad \text{then} \quad y = z$$

Note, however, that is order to cancel an "x" from both sides of the equation, the "xs" must be on the same side of the "$\star$" symbol: each "x" must be on the left side of the "$\star$," or each "x" must be on the right side of the "$\star$." So:

if $x \star y = z \star x$ then it does not follow that $y = z$

And:

if $y \star x = x \star z$ then it does not follow that $y = z$

This is because every algebra is not commutative. If the algebra you are working with is commutative, then of course you can flip one side around, and then cancel the xs. But if your algebra is not commutative, you cannot flip one side around, and you cannot cancel out the xs.

C.4 Inverses

IF COMBINING TWO ELEMENTS yields an inverse, then those two elements must be inverses of each other.

Suppose we have two elements — it doesn't matter which elements they are, so let's just call them x and y. Suppose also that when we combine these two elements, we get the identity element — it doesn't matter what the identity element is either, so let's just call it e. Hence, suppose we have this:

$$x \star y = e$$

If this is the case, then x and y must be the inverses of each other. That is to say, y must be the inverse of x:

$$y = x^{-1}$$

And, x must be the inverse of y:

$$x = y^{-1}$$

We can prove that this. Let's start with what we are given, namely that combining x and y yields the identity e:

$$x \star y = e$$

We know that any element combined with its inverse is equal to e. So, for example, we know that x combined with x^{-1} is equal to e:

$$x \star y = \underbrace{e}_{\text{this is the same as } x \star x^{-1}}$$

Hence, we can rewrite the equation with $x \star x^{-1}$ on the right hand side, instead of e:

$$x \star y = x \star x^{-1}$$

But now notice that x is applied on both sides of the equation. So, we can use the **cancellation law** to cancel out x:

$$\cancel{x} \star y = \cancel{x} \star x^{-1}$$

Hence, we are left with just y and x^{-1}:

$$y = x^{-1}$$

Hence, we have deduced that y must be the inverse of x. That's one of the things we were trying to prove.

Next, we want to prove that x is the inverse of y. To do that, let's go back to where we started, namely that combining x and y yields the identity e:

$$x \star y = e$$

As before, we know that any element combined with its inverse is equal to e. So, for example, we know that y^{-1} combined with y is equal to e as well:

$$x \quad \star \quad y \quad = \quad \underbrace{e}_{\text{this is the same as } y^{-1} \star y}$$

Hence, we can rewrite the equation with $y^{-1} \star y$ on the right hand side, instead of e:

$$x \quad \star \quad y \quad = \quad y^{-1} \quad \star \quad y$$

But notice now that y is applied to both sides of the equation. So, we can again use the **cancellation law** to cancel out the y on both sides:

$$x \quad \star \quad \cancel{y} \quad = \quad y^{-1} \quad \star \quad \cancel{y}$$

Hence, we are left with just x and y^{-1}:

$$x \quad = \quad y^{-1}$$

So we have also deduced that x must be the inverse of y, just as y must be the inverse of x.

C.5 The Inverse of the Inverse

WHAT IS THE INVERSE OF THE INVERSE? Take any element, it doesn't matter which, so let's just call it x. Then, what is the answer to this:

$$(x^{-1})^{-1} = ??$$

The answer is x. The inverse of the inverse of x is just x:

$$(x^{-1})^{-1} = x$$

We can prove this. We know that x combined with its inverse x^{-1} is equal to the identity. It doesn't matter which element the identity is, so let's just call it e. Hence, we know this:

$$x \qquad \star \qquad x^{-1} \qquad = \qquad e$$

But we know from the last section that if combining two elements yields the identity element, then those two elements must be inverses of each other. Hence, the "x" on the left hand side of the "$\star$" symbol must be the inverse of the "x^{-1}" on the right hand side of the "$\star$":

$$\underbrace{x}_{\text{this must be}} \qquad \star \qquad \underbrace{x^{-1}}_{\text{the inverse of this}} \qquad = \qquad e$$

So "x" is the inverse of "x^{-1}." How do we write "the inverse of x^{-1}"? We just add another superscripted "-1," like this: "$(x^{-1})^{-1}$." Hence:

$$x \qquad = \qquad (x^{-1})^{-1}$$

C.6 Combining Inverses

TAKE TWO ELEMENTS and combine them. It doesn't matter which elements they are, so let's just call them x and y:

$$x \star y$$

Now take the inverse of that result:

$$(x \star y)^{-1}$$

That will evaluate to the same thing as if you combine the inverse of y and the inverse of x:

$$(x \star y)^{-1} = y^{-1} \star x^{-1}$$

In other words, if you combine two elements and take their inverse, that is the same as combining the inverses of those two elements in reverse order. Let's prove this.

First, observe that we are claiming that $y^{-1} \star x^{-1}$ is equal to the inverse of $x \star y$. Well, the inverse of an element is whatever other element you combine it with to get e. So, if $y^{-1} \star x^{-1}$ is the inverse of $x \star y$, then we should be able to combine them to get e, like this:

$$\underbrace{(x \star y)}_{\text{whatever this value is}} \star \underbrace{(y^{-1} \star x^{-1})}_{\text{its alleged inverse}} = \underbrace{e}_{\text{the identity}}$$

So the question is, is this true? Are the two sides of this equation equal? The answer is yes. First, let's move the parentheses around. We can do this because groups are associative, so where the parentheses go doesn't matter:

$$x \star \underbrace{(y \star y^{-1})}_{\text{regroup the parentheses}} \star x^{-1} = e$$

Inside the parentheses, we now have "$y \star y^{-1}$," and we know that any element combined with its own inverse evaluates to "e":

$$x \star \underbrace{(y \star y^{-1})}_{\text{this evaluates to the identity } e} \star x^{-1} = e$$

So, we can rewrite this with "e" instead of "$y \star y^{-1}$":

$$x \star (e) \star x^{-1} = e$$

Let's rearrange the parentheses again:

$$\underbrace{(x \star e)}_{\text{regroup the parentheses}} \star x^{-1} = e$$

Inside these parentheses, we have "$x \star e$." Combining any element x with the identity e just yields that same element x:

$$\underbrace{(x \star e)}_{\text{this evaluates to } x} \star x^{-1} = e$$

So, we can replace "$x \star e$" with "x":

$$x \quad \star \quad x^{-1} \quad = \quad e$$

Now, on the left side of this equation, we are combining x and its inverse x^{-1}. And any element combined with its inverse evaluates to the identity e:

$$\underbrace{x \quad \star \quad x^{-1}}_{\text{this evaluates to } e} \quad = \quad e$$

So, it turns out that the left side of this equation does indeed evaluate to e, just like the right side of the equation. Thus, we have shown that it is in fact true that "$x \star y$" and "$y^{-1} \star x^{-1}$" are inverses, since combining them evaluates to e. Hence, we can say that the inverse of "$x \star y$" — that is to say, "$(x \star y)^{-1}$" — is "$y^{-1} \star x^{-1}$":

$$(x \star y)^{-1} = y^{-1} \star x^{-1}$$

And that is what we set out to prove.

C.7 Summary

IN THIS APPENDIX, we looked at some of the basic algebra tricks many are familiar with, and we showed why they work, by proving them.

BIBLIOGRAPHY

Baumslag, B. and B. Chandler (1968). *Group Theory* (Schaum's Outline Series). New York: McGRaw-Hill Book Company.

Benjamin, Arthur, Gary Chartrand, and Ping Zhang (2015). *The Fascinating World of Graph Theory*. Princeton, NJ: Princeton University Press.

Blackburn, Patrick and Johan Bos (2005). *Representation and Inference for Natural Language: A First Course in Computational Semantics*. Stanford, CA: CSLI Publications.

Boolos, George, John Burgess, and Richard Jeffrey (2002). *Computability and Logic*, 4th edition. Cambridge: Cambridge University Press.

Burger, Edward and Michael Starbird (2010). *The Heart of Mathematics: An Invitation to Effective Thinking*, 3rd ed. Hoboken, NJ: John Wiley & Sons, Inc.

Chartrand, Gary (1977). *Introductory Graph Theory*. New York:

Dover Publications, Inc. Reprint of the Prindle, Weber, & Schmidt edition.

Cummings, Jay (2020). *Proofs: A Long-Form Mathematics Textbook*. Self-published.

Cummings, Jay (2018). *Real Analysis: A Long-Form Mathematics Textbook*, 2nd ed. Self-published.

Davey, B. A. and H. A. Priestley (1990). *Introduction to Lattices and Order*. Cambridge: Cambridge University Press.

Devlin, Keith (2012). *Introduction to Mathematical Thinking*. Petaluma, CA: Self-published by Keith Devlin.

Dos Reis, Laura L. and Anthony J. Dos Reis (2017). *Abstract Algebra: A Student-Friendly Approach*. Self-published.

Dowty, David R., Robert E. Wall, and Stanley Peters (1981). *Introduction to Montague Semantics*. Boston: Kluwer Academic Publishers.

Enderton, Herbert. *Elements of Set Theory*. Boston: Academic Press.

Flegg, H. Graham (1974). *From Geometry to Topology*. New York: Dover Publications, Inc. Reprint of the Crane, Russak, and Co. edition.

Fournier, Jean-Claude (2009). *Graph Theory and Applications*. Hoboken, NJ: Wiley.

Gould, Ronald (2012). *Graph Theory*. New York: Dover Publications, Inc. Reprint of the 1988 Benjamin-Cummings edition.

Halmos, Paul. *Naive Set Theory*. New York: Dover Publications, Inc. Reprint of the 1960 Van Nostrand edition.

Hartsfield, Nora and Gerhard Ringel (1990). *Pearls of Graph Theory: A Comprehensive Introduction*. Boston: Academic Press, Inc.

Jongsma, Calvin (2019). *Introduction to Discrete Mathematics via Logic and Proof*. Cham, Switzerland: Springer.

Madden, Daniel J. and Jason A. Aubrey (2017). *An Introduction to Proof through Real Analysis*. Hoboken, NJ: Wiley.

Pinter, Charles (2014). *A Book of Set Theory*. New York: Dover Publications, Inc. Reprint of the 1971 Addison-Wesley edition.

Pinter, Charles (2010). *A Book of Abstract Algebra*, 2nd edition. New York: Dover Publications, Inc. Reprint of the 1982 McGraw-Hill edition (also reprinted by Dover in 1990).

Saracino, Dan (2008). *Abstract Algebra*, 2nd edition. Long Grove, IL: Waveland Press, Inc.

Sawyer, W. W. (1982). *Prelude to Mathematics*. New York: Dover Publications, Inc. An enlarged version of the 1955 Penguin edition.

Sawyer, W. W. (2007). *Mathematician's Delight*. New York: Dover Publications, Inc. Reprint of the 1943 Penguin edition (also reprinted by Dover in 1991.

Sawyer, W. W. (2018). *A Concrete Approach to Abstract Algebra*. New York: Dover Publications, Inc. Reprint of the 1959 W. H. Freeman and Company edition (also reprinted by Dover in 1978).

Schmidt, O. U. (1966). *Abstract Theory of Groups*, trans. by Fred Holling and J. B. Roberts, ed. by J. B. Roberts. San Francisco, CA: W. H. Freeman and Company.

Smullyan, Raymond and Melvin Fitting (2010). *Set Theory and the Continuum Problem*. New York: Dover Publications, Inc. Reprint of the 1996 Oxford University Press edition.

Steinhart, Eric (2018). *More Precisely: The Math You Need to Do Philosophy*, 2nd ed. Peterborough, Ontario: Broadview Press.

Stewart, Ian (1995). *Concepts of Modern Mathematics*. New York: Dover Publications, Inc. Reprint of the 1981 Penguin edition.

Stewart, Ian and David Tall (2015). *The Foundations of Mathematics*, 2nd. edition. Oxford: Oxford University Press.

Teller, Paul (1989). *A Modern Formal Logic Primer*, 2 volumes. Upper Saddle River, NJ: Prentice Hall.

Trudeau, Richard (1993). *Introduction to Graph Theory*. New York: Dover Publications, Inc. Reprint of the 1976 Kent State University Press edition.

Velleman, Daniel (2019). *How to Prove It: A Structured Approach*, 3rd edition. Cambridge: Cambridge University Press.

Warner, Seth (1990). *Modern Algebra*, two volumes bound as one. New York: Dover Publications, Inc. Reprint of the 1965 Prentice-Hall 2-volume edition.

Warner, Steve (2019). *Topology for Beginners*. Self-published.

Wilder, Raymond (2012). *Introduction to the Foundations of Mathematics*, 2nd edition. New York: Dover Publications, Inc. Reprint of the 1965 John Wiley & Sons edition (also reprinted by Dover in 1983).

Wolf, Robert (1998). *Proof, Logic, and Conjecture: The Mathematician's Toolbox*. New York: W.H. Freeman and Company.

Zach, Richard (2019). *Sets, Logic, Computation: An Open Introduction to Metalogic*, Fall 2019 edition. Self-published.