M. ANDREW MOSHIER

CONTEMPORARY DISCRETE MATHEMATICS

M&H PUBLISHING
Copyright © 2019 M. Andrew Moshier

PUBLISHED BY M&H PUBLISHING

MANDREWMOSHIER.GITHUB.IO/LNDM/
Contents

What is this about? 4

I Natural Numbers and Induction 9

1 The Natural Numbers 11

2 Arithmetic 18

3 Laws of Arithmetic 22

4 Examples of Recursion and Induction 31

5 Ordering and Strong Induction 44

II Sets, Functions, Relations and Proofs 52

6 Sets, Functions, Relations 57

7 Functions in More Detail 71
8 Relations and Powersets  92

9 The Set of Natural Numbers  97

10 Lists  104

III Applications  115

11 Minimum and Maximum  116

12 Divisibility: Ordering the Natural Numbers by Multiplication  120

13 Greatest Common Divisors and Least Common Multiples  128

14 Prime Numbers  137
What is this about?

Discrete mathematics consists of many individual topics that, imprecisely, contrast with continuous mathematics, e.g., topics like calculus, real analysis, differential equations, and so on. But a sharp contrast between discrete and continuous mathematics is mainly a convenience. We put some topics in one basket, others in a different basket, but things are more connected than that.

Discrete mathematics texts tend to be disorganized, featuring disparate topics that start looking like an Island of Misfit Math — stuff that simply does not fit anywhere else. In this text, I take the view that discrete mathematics comprises a coherent set of ideas centered around some key themes:

- Discrete mathematics is mainly about structure and information. Numbers play an important role, but mainly because different sorts of numbers convey different sorts of information. For example, the number of students in a class conveys some information about the class. Or the area under a certain curve conveys some information about the curve.
- The natural numbers and closely related inductive structures are central because they provide a bridge between mathematics and computation.
- Mathematical data is “typed”. Natural numbers constitute a type of data; real numbers constitute a different type of data; polynomials with real coefficients constitute yet another type of data, and so on.
- Algebraic (equational and inequational) reasoning provides key insights about how structures can be related.
- Mathematics itself is computational — at least it is extremely productive to think that way as much as possible.
- Abstraction and use of analogy is a crucial means of discovering new ideas, and of tying what we know together.

In these lecture notes, we deal head on with mathematics as the study of abstract structure. This can be frustrating at first because the concrete applications are not always obvious. The pay off comes when we see that generalization from particulars leads to much wider applications than we could have anticipated.

For example, \(154133 + 11^{11}\) (a number with more than 100 trillion digits) is not divisible by 11. These two numbers are far too big to calculate explicitly as a numeral. We can not actually hope to perform the exponentiations and addition, and check whether the result is divisible by 11. Yet, we are completely confident that if we could perform the calculations, the result would be a number that is not divisible by 11.

We should be able to convince ourselves that \(m^n\) is always divisible by \(m\) whenever \(n\) is a positive integer, not just by taking that for granted. Likewise, we can check (by performing a long division) that
154133 is not evenly divisible by 11. Finally, we should also be able to convince ourselves that if \( m \) is not divisible by some number (let’s call it \( p \)) and \( n \) is divisible by \( p \), then \( m + n \) is not divisible by \( p \). So, we can reason about a particular example indirectly by working about more general principles.

Figuring out how to convince someone that something like this must true is actually the main activity of mathematics.

**Persuasion**

Mathematicians spend a lot of time thinking about what is, or isn’t true. But that would be pretty useless, if they did not then communicate the result of their thinking. To actually put mathematics to work, you must learn how to persuade someone else to accept a proposed solution to a problem. It is not enough to know how to solve problems.

The standards for persuasion in mathematics are very high. We do not settle for “preponderance of evidence” or “beyond reasonable doubt”, or “that seems reasonable.” We aim for “beyond any doubt.” In most human endeavours (including in the sciences), that standard would be paralyzing, but in mathematics, where abstract structure is the object of investigation, it is not only within reach, but it is the standard that makes sense.

In mathematics, a proof is a convincing argument. So it falls into the genre of persuasive writing. This point of view is critical for understanding how mathematicians work.

The emphasis on writing is important. After all, no one cares what I think, unless I manage to convince him or her that my thinking is on track.

Most exercises in these notes are not of the sort: “Solve for \( x \): \( 4x - 3 = x^2 \).” Rather, an exercise might say “Show that the equation \( 4x - 3 = x^2 \) has two solutions.” The former would tempt you simply to write (with a box around the answer, of course) something like \( x = 1, 3 \). The latter asks you to convince the reader that these two values are the only solutions.

Particularly in the first part of the text, I will emphasize fairly formulaic writing. This is meant to give you practice with precise mathematical writing. Think of it as practicing basic scales and arpeggios.

**Roughly, an Outline**

These notes start by investigating two ideas that are fundamental to all of mathematics.

- First, we look carefully at the idea of counting things. You might think there is not much to say about something you have been doing reliably for most of your life. But arithmetic (addition and multiplication, for example) is tightly related to counting. Understanding arithmetic relies on understanding how counting works.

- Second is the idea of putting things into lists. Again, you have been doing this a long time. You write down shopping lists, you take two lists and merge them into one list, and so on. Lists are crucial for everything in mathematics, even how we write a number in base ten. After all, what exactly does 3429 mean? A base ten numeral is formed by putting symbols (digits) next to each other into a certain order. So 3429 is not the same numeral as 3492. Lists provide a general way to think about things like this.

After looking at the basics of counting and of lists, we turn to the foundational language of contemporary
mathematics: sets, functions and relations. Informally, a set is simply a collection of things, such as \( \mathbb{N} \) — the collection of natural numbers. A function is a correlation of the things in one set with things in another set. A familiar example is \( f(x) = x^2 \) that correlates with any real number \( x \), the square \( x^2 \). A relation is what is says it is: a way to relate things in one set to things in another.

To make sense of sets, functions and relations, we need to consider some basic questions:

- What does it mean say two sets are equal?
- What does it mean to say two functions are equal?
- What does it mean to say two relations are equal?
- How can we specify or “build” particular sets?
- How can we specify or “build” particular functions?
- How can we specify or “build” particular relations?

The middle part of this text concerns these questions.

In the last part of the text, we investigate a variety of useful mathematical ideas using the techniques developed in the first two parts. The emphasis is mostly on practical ideas that either emerge from computing, or are directly useful there.

Some basics

Before we launch into mathematics proper, we need to sort out some basic notation and discuss how to read this text. As usual, we typically use single letters, for example, in \( x^2 - 2x + 1 \), as placeholders for values. These are frequently called “variables”, but there may or may not be anything “varying” about the idea. For example, in the equation \( 0 = x^2 - 2x + 1 \), there is exactly one real number (1) that can be the value of \( x \).

There is nothing special about the name ‘\( x \)’. I might as well have written \( y^2 - 2y + 1 \) or \( a^2 - 2a + 1 \). The difference is only apparent when we make assumptions about the variables. For example, look at these two:

- “Suppose \( x \) is a positive real number. Consider \( x^2 + 3x + 1 \).”
- “Suppose \( x \) is a positive real number. Consider \( y^2 + 3y + 1 \).”

Clearly, they mean different things. In the first one, the polynomial expression is guaranteed to be positive. In the second one, the polynomial has nothing to do with \( x \). Contrast this with these two:

- “Suppose \( x \) is a positive real number. Consider \( x^2 + 3x + 1 \).”
- “Suppose \( y \) is a positive real number. Consider \( y^2 + 3y + 1 \).”

Evidently, these two really do mean the same thing. It is important to keep details like this in mind. To understand how a variable should be interpreted look for the context in which it is first mentioned.

Frequently, we use letters in certain parts of the alphabet to refer to specific types of values. For example, I will mostly use \( m, n \) and \( p \) to refer to natural numbers, and keep using \( x, y \) and \( z \) for generic values (not from a particular type of value). Usually \( f, g, h \) stand for functions. Later on, other types of entities will be needed. I will use Greek letters for some things. So it is in your interest to be familiar with the Greek alphabet — at least, some commonly used letters.
Sometimes, a datum should have no meaning on its own. Symbols are just names with no interpretation. I will typically write a symbol in a type face like so: \textit{waffles}. Thus $x$ is a symbol, whereas $x$ is a variable. In handwritten text, a symbol might be indicated by underlining.

Two other type faces will come in handy: \texttt{san serif} and \texttt{SMALL CAPS}. Usage will not make much sense now. I’ll wait to explain how I intend to use them when the time is right.

Parentheses are used in mathematics in several distinct ways. This, unfortunately, can be a source of misunderstanding. Here are some of the ways they are used.

- Function application. We frequently write $f(5)$ to mean “apply the function $f$ to the value 5.”
- Pairs, triples and so on: $(4,5)$ means “the pair consisting of 4 then 5.”
- Ranges. In some situations, the notation $(0,1)$ means “the collection of real numbers strictly between 0 and 1.
- Grouped sub-expressions. In an expression such $a + b \cdot c$, you know to evaluate $b \cdot c$ before added $a$ to the result. But to get the result of multiplying $a + b$ with $c$, you would write $(a + b) \cdot c$.

Usually, these different uses of parentheses will not cause any confusion because the context will make it clear what is meant. I just caution you to pay close attention.

Throughout the text, I refer to certain parts as \textit{algorithms} or \textit{definitions}. The distinction is not precise. The idea is that an algorithm is meant to convey a precise way of calculating something; a definition is meant to convey a way to talk about something. For example, addition of two natural numbers is presented as an algorithm because there is a precise sequence of concrete steps to take when adding. But $\leq$ (the relation of being less than or equal to) is presented as a definition. Even though there may be a calculation to determine whether $m \leq n$, the calculation is not the main idea. There are no concrete steps in mind for determining whether $m \leq n$ is true.

This usage of the word algorithm is not meant to be formal. An algorithm in the text is not a program in a particular programming language. Rather, an algorithm is just a precise description of a procedure that might be carried out by hand or by computer.
Part I

Natural Numbers and Induction
Our earliest mathematical experience is learning to count. Arithmetic on the natural numbers, simple as it seems, exhibits many of the features of higher contemporary mathematics that concern us throughout this course. By starting with a careful look at arithmetic, you prepare yourself for what comes in this course and in later mathematics.

An important technique for reasoning about natural numbers and many other mathematical structures is called induction. Induction is the main theme of Part I, and is used throughout the rest of the text.
1

The Natural Numbers

The natural numbers have to do with counting. In this chapter, the structure of natural numbers is the topic. The natural numbers constitute the most important data type of mathematics and computing. As the 19th century mathematician Kronecker quipped “Die natürlichen Zahlen hat der liebe Gott gemacht, alles andere ist Menschenwerk” (the natural numbers were made by God, the rest is man’s handiwork).

To hone in on that structure, we look at similar structures that fail to capture basic aspects of counting. These bogus structures can be ruled out by being precise about how counting behaves. The result is a system of easily remembered postulates that distinguish the structure of natural numbers from all others.

1.1 The Basic Picture

A question “how many X’s are there?” is distinct from “How much x is there?”. The latter sort of question could be answered with $\frac{1}{2}$ (as long as we know how we are measuring). On the other hand, the answer to a “how many” question could be 1, 2, 3 and so on. The answer could also be 0 (for example, “how many math professors own unicorns?”). On the other hand, “how many?” cannot be answered with $\frac{1}{2}$, π, −5 and so on.

Natural numbers may be pictured like stepping stones as in Figure 1.1.

![Figure 1.1: A picture of the natural numbers](image)

Not all “stepping stone” pictures are acceptable as pictures of the natural numbers. Figures 1.2, 1.3 and 1.4 illustrate three ways not to
picture the natural numbers.

![Diagram of a path with forks and arrows]

Figure 1.2: Forks in the path

![Diagram of a path with no arrows]

Figure 1.3: Nowhere to go

![Diagram of a path with arrows]

Figure 1.4: Nowhere to start

Examples of how things can go wrong play an important role in math. They are called counter-examples.

These incorrect pictures can be ruled out by explaining the basic vocabulary of counting.
Vocabulary 1: Basic Vocabulary of Natural Numbers

The natural numbers have the following features.

• There is a special natural number, zero, denoted by 0.

• For any natural number \( n \), there is a unique next natural number, called the successor of \( n \). In these notes, the successor of \( n \) is denoted by \( n \rightarrow \).

The collection of natural numbers is denoted by \( \mathbb{N} \). To indicate that a variable \( n \) is intended to range over natural numbers, we write \( n \in \mathbb{N} \). So for example, saying “Every \( n \in \mathbb{N} \) is either odd or even” is the same as saying “Every natural number \( n \) is either odd or even.”

According to Vocabulary 1, expressions like 0, 0\( \rightarrow \), 0\( \rightarrow \rightarrow \), 0\( \rightarrow \rightarrow \rightarrow \) denote natural numbers. Of course, we usually abbreviate these by writing 0, 1, 2, 3. But the characters 1, 2, 3, etc., are not related to each other. Vocabulary 1 makes it clear that 0\( \rightarrow \) is meant to be the number after 0; 0\( \rightarrow \rightarrow \), the number after that, and so on. We will want to be able to switch between the familiar “decimal” notation and “successor” notation whenever it is convenient. Assuming the plus sign is part of an augmented vocabulary, \( 2 + 2 = 4 \) could be written as

\[ 0\rightarrow\rightarrow + 0\rightarrow\rightarrow = 0\rightarrow\rightarrow\rightarrow\rightarrow. \]

The next exercises emphasize the switch back and forth.
1.2 Narrowing to actual counting

Figures 1.5 and 1.6 illustrate correct pictures of Vocabulary 1. Obviously, they are not pictures of the natural numbers. It is not enough just to spell out carefully how to talk about counting. We also need to explain what is not allowed. Postulates play that role.

Figure 1.5 is flawed because 0 has a predecessor: a value n satisfying n\(\Leftarrow\) = 0. Figure 1.6 is flawed because an element has two distinct predecessors: 0\(\Leftarrow\) = 0\(\Leftarrow\)\(\Leftarrow\)\(\Leftarrow\)\(\Leftarrow\).

We can rule these problems out by postulating behavior of 0 and of successors.
**POSTULATE 1: Nothing Precedes 0**

For every natural number \( n \), it is the case that \( n \preceq \neq 0 \).

**POSTULATE 2: Predecessors are Unique**

For any natural numbers \( m \) and \( n \), if \( m \preceq = n \preceq \) then \( m = n \).

Suppose you are told that “\( m \) has a predecessor.” Then you know \( m = k \preceq \) for some \( k \). By the previous two postulates, you immediately know two more things. First, \( m \neq 0 \). Second, \( k \) is unique. That is, we can speak about the predecessor of \( m \). This suggests that we can think of those natural numbers with predecessor specially. Let us make that official.

**DEFINITION 1: Positive Natural Numbers**

We say that a natural number \( m \) is *positive* if \( m \) has a predecessor. In particular, \( 0 \) is not positive. Also, following the notation \( \mathbb{N} \) for the collection of all natural numbers, we write \( \mathbb{N}^+ \) for the collection of all positive natural numbers.

For \( m \in \mathbb{N}^+ \), the *predecessor* of \( m \) is the unique \( k \in \mathbb{N} \) satisfying \( m = k \preceq \). We will write \( \text{pred}(m) \) for this. In other words, for each \( m \in \mathbb{N}^+ \),

\[
  m = \text{pred}(m) \preceq
\]

and for each \( n \in \mathbb{N} \),

\[
  n = \text{pred}(n \preceq).
\]

The postulates eliminate Figures 1.5, 1.6 and similar bad pictures. But there is still a problem, illustrated in Figure 1.7. The circle labelled \( \star \) has a predecessor because \( \star \preceq = \star \). But it is not clear that it makes sense to call it *positive*. 
The picture in Figure 1.7 uses the basic vocabulary correctly, and satisfies the first two postulates. Yet, it is not a picture of natural numbers because it has “extra stuff” in it. In this case, ⋆ is “extra stuff,” but we could have concocted more complicated examples. The challenge is to rule out any possible “extra stuff”.

Were we to erase the circle labelled ⋆ and any arrows leading to and from it, the remaining part of Figure 1.7 would still live up to the basic vocabulary of natural numbers (Vocabulary 1). This is exactly what makes for “extra stuff”: elements that can be removed without doing any harm to our ability to talk about all natural numbers. Clearly, we can not remove 0 because the basic vocabulary requires it. And we can not remove 0↷ because the basic vocabulary requires 0 to have a successor. Likewise, we cannot remove 0↷↷ because the basic vocabulary requires 0↷ to have a successor. And so on. So the values that can be reached by a sequence of steps (that is, by counting) from 0 must stay. Anything else would be extra. This leads to our last postulate.

\[\text{POSTULATE 3: The Axiom of Induction}\]

No natural numbers can be removed without violating Vocabulary 1.

Believe it or not, the vocabulary and three postulates we have stated here completely characterize the standard picture of the natural numbers. In other words, any picture that satisfies these will look the same. A rigorous proof of this is possible, but not necessary for now.

With this axiom, the term positive actually means what we expect it to mean: \(m\) is positive if and only if \(m\) is strictly greater than 0. To make that claim precise we need to know what strictly greater than should mean. This is a topic discussed in Chapter 11.
Exercises

9. Each of the following pictures violates the vocabulary or one or more of our postulates. For each, explain what is violated.

(a) 

(b) 

(c) 

(d) 

10. I have in mind a picture for Vocabulary 1 and Postulates 1 and 2. Furthermore, in the picture, I have in mind an element \( n \) for which (a) \( n \neq 0 \) and (b) \( n \) has no predecessor (that is, \( n \neq k^\prec \) for every natural number \( k \)). Convince me that the picture fails to satisfy Postulate 3.

11. Draw three different pictures of situations that satisfy all the postulates except that they fail Postulate 1. So there will be an arrow from a bubble into the bubble labelled 0. The result must satisfy all other postulates including the Axiom of Induction.

Exercise 10 shows that in the natural numbers, if \( n \neq 0 \) then \( n = k^\prec \) for some \( k \). In other words, every \( n \in \mathbb{N} \) falls into exactly one of two mutually exclusive cases: either \( n = 0 \) or \( n \) has a predecessor.
2

Arithmetic

Adding and multiplying arise from counting. In this chapter, we explore how to define them computationally, purely in terms of counting.

2.1 Basic Arithmetic Operations

Addition works by counting ahead. For example, to add 4 + 5, start with 4 and then count five more. This is how you first learned to add. Multiplication works by counting a number of additions. To multiply 2 · 3, add 2 three times: 2 + 2 + 2. The following capture the idea.

Algorithm 1: Addition

The sum of \( m \in \mathbb{N} \) and \( n \in \mathbb{N} \) is a natural number \( m + n \), calculated by the following:

\[
\begin{align*}
m + 0 &= m \\
m + k^\bowtie &= (m + k)^\bowtie \quad \text{for any } k
\end{align*}
\]
Algorithm 2: Multiplication

The product of \( m \in \mathbb{N} \) and \( n \in \mathbb{N} \) is a natural number \( m \cdot n \), calculated by the following:

\[
\begin{align*}
m \cdot 0 &= 0 \\
m \cdot k^\sim &= m + (m \cdot k) & \text{for any } k
\end{align*}
\]

An algorithm is a precise specification for how to calculate something. The foregoing are both algorithms in so far as they spell out how to calculate \( m + n \) and \( m \cdot n \) for any \( m \) and \( n \).

The vocabulary and postulates of natural numbers ensure that there are indeed unique operations \( + \) and \( \cdot \) that satisfy the equations. A detailed proof of this fact is not illuminating right now. We will be better off proving that recursive algorithms like this always work. Then sum and product will turn out just to be particular instances of a general method for specifying algorithms. We will return to this point in Chapter 9.

In the next chapter, we will establish a proof pattern called “simple arithmetic induction” generalizing this argument. This is one of the mainstays of mathematics and computing.

Induction also shows that the equations for sums and products define operations.

Example 1:

Calculate \( 4 + 3 \):

\[
\begin{align*}
3 + 4 &= 0^\sim + 4 & [3 \text{ abbreviates } 0^\sim + 4] \\
&= (0^\sim + 4)^\sim & [\text{k}^\sim + n = (k + n)^\sim] \\
&= (0^\sim + 4)^\sim & [\text{Same reason}] \\
&= (0 + 4)^\sim & [\text{Same reason}] \\
&= 4^\sim & [0 + n = n] \\
&= (0^\sim + n)^\sim & [4 \text{ abbreviates } 0^\sim + n] \\
&= 0^\sim + n & [\text{Remove unneeded parentheses}] \\
&= 7 & [7 \text{ abbreviates } 0^\sim + n]
\end{align*}
\]

In this text, I will always denote multiplication with an explicit \( \cdot \). Though it is conventional to write things like \( 2m \) instead of \( 2 \cdot m \), I keep the multiplication sign to make things clearer.
A product can be calculated similarly. Consider $3 \cdot 2$.

\[
3 \cdot 2 = 3 \cdot 0 \xrightarrow{\text{[2 abbreviates } 0 \xrightarrow{\text{]}}}
= 3 + (3 \cdot 0) \xrightarrow{\text{[Same reason]}}
= 3 + (3 + 0) \xrightarrow{\text{[m \cdot 0 = 0]}}
= 3 + 3 \xrightarrow{\text{[m + 0 = m]}}
= 3 + 0 \xrightarrow{\text{[3 abbreviates } 0 \xrightarrow{\text{]}}}
= (3 + 0) \xrightarrow{\text{[Same reason]}}
= (3 + 0) \xrightarrow{\text{[Same reason]}}
= 3 \xrightarrow{\text{[m + 0 = m]}}
= (0 + 3) \xrightarrow{\text{[3 abbreviates } 0 \xrightarrow{\text{]}}}
= 0 + 3 \xrightarrow{\text{[Remove unnecessary parentheses]}}
= 6 \xrightarrow{\text{[6 abbreviates } 0 \xrightarrow{\text{]}}}
\]

We certainly will not want to calculate this way in real life. Even a simple calculation like $3 \cdot 2 = 6$ took thirteen steps. But these examples show how addition and multiplication are built mechanistically from counting.

Exercises

12. Calculate these sums, following the previous example to write each step of your calculation explicitly. Include the reason for each step (as in the previous example). Take care to lay out the chain of equalities correctly, and do not skip any steps.

(a) $2 + 4$
(b) $4 + 2$
(c) $3 + (3 + 1)$
(d) $(3 + 3) + 1$
(e) $0 + 3$

13. Notice that it takes more steps to calculate $2 + 4$ than $4 + 2$, even though they produce the same answer. Explain why.

14. Calculate the following values, writing each step explicitly.
(a) \( 2 \cdot 3 \)
(b) \( 0 \cdot 2 \)
(c) \( 2 \cdot (2 \cdot 2) \)
(d) \( 3 \cdot (2 + 1) \)
(e) \( (3 \cdot 2) + (3 \cdot 1) \)

15. Write a definition of exponentiation via equations for \( m^0 \) and for \( m^k \). Follow the pattern of definition for addition and multiplication.
3

Laws of Arithmetic

YOU DO NOT NEED TO CALCULATE anything to know that

\[ 335 \cdot (23 + 17) = 335 \cdot 23 + 335 \cdot 17. \]

This is because you know that multiplication always \textit{distributes over} addition. This is a law of arithmetic. You can think of several other laws we use all the time, such as commutativity — \( m + n = n + m \) — and associativity — \( m + (n + p) = (m + n) + p \).

We can prove that these laws hold by virtue of the structure of counting and the definitions of addition and multiplication. In this chapter, we see that the Axiom of Induction gives rise to an important pattern for proving facts about natural numbers. Later we generalize the technique to other kinds of data.

The following table summarizes several useful laws of arithmetic on the natural numbers. Most of them will be familiar to you.

---

LAW 1: Basic Laws of Arithmetic

For any natural numbers, \( m, n \) and \( p \):

---

CHAPTER GOALS

Remind ourselves of the basic laws of addition and multiplication. Establish the pattern of proof by simple arithmetic induction. Illustrate how to use the pattern by proving that these laws hold.

We use the names of these laws frequently. Many of them show up in non-numerical contexts as well and have common, familiar names. So you will do yourself a favor by memorizing them. A few of them will be more obscure to you right now, but they are all important for actually using arithmetic.

This table is organized to emphasize similarities between addition and multiplication. Pay attention to that.
The Law of Case Distinction was the subject of Chapter ?? Exercise 10. Go back and look at that exercise again.

3.1 Inductive Proofs

Suppose we wish to prove that every natural number has some property. For example, we wish to prove that $0 + m = m$ for all natural numbers $m$. This is obviously true. But why? Our definition of $+$ says that $m + 0 = m$, not the other way around. So something needs to be explained. We could try saying “addition is commutative, so $0 + m = m + 0$ and by the definition $m + 0 = m$”. But now we have another problem. How do we know that addition is commutative? Again, the defining equations for addition do not explicitly tell us this.

The Axiom of Induction holds the key to proving that some fact is true for all natural numbers. The idea is simple, but will take some getting used to. Back to the example of $0 + m = m$. We could simply try to check for all $m$ whether this is true. First, is $0 + 0 = 0$ true? Yes. Is $0 + 0^\vdash = 0^\vdash$? Now we need to do a bit of calculating. $0 + 0^\vdash = (0 + 0)^\vdash$ according to the definition of addition. But we just noted that $0 + 0 = 0$. Now is $0 + 0^\vdash = 0^\vdash$? Again, we will need to do some calculating. Evidently, “just check everything” is not going to work because we will never complete the task.
On the other hand, think about the natural numbers that do happen to satisfy $0 + m = m$. If just these natural numbers provide us with a model of Vocabulary 1, then the Axiom of Induction asserts that we can not remove any natural numbers from the ones that happen to satisfy $0 + m = m$. So all natural numbers must.

Let us see how it works in practice. Say that a natural number $m$ is nice if it has the property $0 + m = m$. We want to prove that all natural numbers are nice. What we really do is show that the nice natural numbers form a picture of Vocabulary 1.

- Evidently, $0$ is nice because $0 + 0 = 0$. That is part of the algorithm for addition.
- Suppose $k$ is a nice natural number.
- For the nice natural numbers to form a model of Vocabulary 1, it must be the case that $k^\succ$ is also nice. So we check:

$$0 + k^\succ = (0 + k)^\succ \quad \text{[By the addition of algorithm.]}$$

$$= k^\succ \quad \text{[Because we suppose } k \text{ is nice]}$$

Putting these together, we see that the nice natural numbers form the characteristic “stepping stone” picture: zero is nice, any successor of a nice number is nice. So the Axiom of Induction ensures that all natural numbers are nice.

A proof employing the Axiom of Induction in this way is called a proof by simple arithmetic induction, or just a proof by induction, for short. We will see more general forms of induction later.
To make inductive proofs easier to understand, we often write them in the three-step outline suggested above. To prove that all natural numbers are “nice”:

**Basis** Prove that 0 is nice.

**Inductive Hypothesis** Assume that k is nice for some (unspecified) \( k \in \mathbb{N} \).

**Inductive Step** Prove that \( k \rightarrow \) is also nice.

From these, the nice numbers constitute a model of Vocabulary 1. So by the Axiom of Induction, all natural numbers are nice.

The familiar laws of arithmetic are now provable by induction. We do not have to accept them “from authority.” We can check for ourselves why they must be true.

**Proposition 1:** Addition is associative.

For any natural numbers \( m, n \) and \( p \),

\[
m + (n + p) = (m + n) + p.
\]

**Proof:** Suppose that \( m \) and \( n \) are fixed natural numbers (not known to us). We prove the result by induction on \( p \).

**Basis** The goal of the basis is the show that \( m + (n + 0) = (m + n) + 0 \). But \( m + (n + 0) = m + n = (m + n) + 0 \) due to the definition of +.

**Inductive Hypothesis** Assume that \( m + (n + k) = (m + n) + k \) for some \( k \in \mathbb{N} \).

**Inductive Step** The goal is to show that \( m + (n + k \rightarrow) = (m + n) + k \rightarrow \).

\[
m + (n + k \rightarrow) = m + (n + k)^\rightarrow \quad [\text{Def. of +}]
\]
\[
= (m + (n + k))^\rightarrow \quad [\text{Same reason}]
\]
\[
= ((m + n) + k)^\rightarrow \quad [\text{Inductive Hypothesis}]
\]
\[
= (m + n) + k^\rightarrow \quad [\text{Def. of +}]
\]

Therefore (by the Axiom of Induction), \( m + (n + p) = (m + n) + p \) holds for all natural numbers \( p \). Since this argument does not depend on
any extra assumptions about \( m \) and \( n \), it holds for all natural numbers \( m \) and \( n \). □

In the remainder of this section, we illustrate the technique of simple arithmetic induction by proving other laws of arithmetic.

**Proposition 2:** \( 0 \) is the identity for addition.

*For any \( m \), it is the case that \( m + 0 = m \) and \( m = 0 + m \).*

**Proof:** The first equality is true by the definition of \(+\). The second equality, \( m = 0 + m \), was proved above as an introduction to induction. □

To prove that addition is commutative, we need another fact about how successor and addition interact.

**Lemma 1:** Successors migrate

*For any \( m \) and \( n \), \( m + n^\wedge = m^\wedge + n \).*

**Proof:** We prove the claim by induction on \( n \). Suppose \( m \) is some fixed natural number.

**Basis** The goal is to show that \( m + 0^\wedge = m^\wedge + 0 \).

\[
\begin{align*}
m + 0^\wedge &= (m + 0)^\wedge & \text{[Def. of +]} \\
&= m^\wedge & \text{[Def. of +]} \\
&= m^\wedge + 0 & \text{[Def. of +]}
\end{align*}
\]

**Inductive Hypothesis** Suppose \( m + k^\wedge = m^\wedge + k \) for some \( k \in \mathbb{N} \).

**Inductive Step** The goal is to show that \( m + k^\wedge + n = m^\wedge + k^\wedge + n \).

\[
\begin{align*}
m + k^\wedge + n &= (m + k^\wedge)^\wedge + n & \text{Def. of +} \\
&= (m^\wedge + k^\wedge)^\wedge + n & \text{[Inductive Hypothesis]} \\
&= m^\wedge + k^\wedge + n & \text{[Def. of +]}
\end{align*}
\]

So \( (m + n)^\wedge = m^\wedge + n \) for all \( n \). Because the proof does not depend on any assumption about \( m \), it is valid for all \( m \). □

Roughly speaking this lemma, and the definition of \(+\), permit us to

Mathematicians typically use the symbol □ as a punctuation mark to indicate the end of a proof.

Because we are currently discussing natural numbers and nothing else, it is safe to write “For any \( m \)” instead of “For any natural number \( m \).”

Mathematicians use the word *lemma* to indicate that a certain fact is needed to make other proofs easier. It is sort of like a subroutine.
move $\mapsto$ anywhere within an addition: $m \mapsto + n = (m + n) \mapsto = m + n \mapsto$.
So we are free to move a successor “out of the way” whenever we need to. The next proof illustrates the point.

**Proposition 3:** Addition is commutative.

For any natural numbers $m$ and $n$,

$$m + n = n + m$$

**Proof:** We need to show that $m + n = n + m$ for all $m$ and $n$. This time, the proof is by induction on $m$. Fix a value for $n \in \mathbb{N}$.

**Basis** The goal is to show that $0 + n = n + 0$. But $0 + n = n = n + 0$ holds because of Proposition 2 and the definition of $+$. 

**Inductive Hypothesis** Assume that $k + n = n + k$ for some $k$.

**Inductive Step** The goal is to show that $k \mapsto + n = n + k \mapsto$.

$$
\begin{align*}
k \mapsto + n &= k + n \mapsto & \text{[Lemma 1]} \\
&= (k + n) \mapsto & \text{[Definition of $+$]} \\
&= (n + k) \mapsto & \text{[Inductive Hypothesis]} \\
&= n + k \mapsto & \text{[Def. of $+$]}
\end{align*}
$$

Therefore, $m + n = n + m$ for all $m$. Since this argument does not depend on any assumptions about $n$, it is valid for all $n$. $\Box$

The next law may be less familiar to you. Roughly, it says that we can “subtract” equals and get equals. But because actual subtraction does not generally make sense for natural numbers ($5 - 7$ means nothing without introducing negative numbers), cancellativity is the best we can do.
PROPOSITION 4: Addition is cancellative.

For any natural numbers \( m, n \) and \( p \), if \( m + p = n + p \) then \( m = n \).

**Proof:** The proof is by induction on \( p \). Assume that \( m \) and \( n \) are some fixed natural numbers.

**Basis** The goal is to show that if \( m + 0 = n + 0 \), then \( m = n \). Suppose \( m + 0 = n + 0 \). Then immediately by definition of \( + \), \( m = n \).

**Inductive Hypothesis** Assume that the following statement is true for some \( k \in \mathbb{N} \): if \( m + k = n + k \) then \( m = n \).

**Inductive Step** The goal is to show that if \( m + k \mapsto = n + k \mapsto \) then \( m = n \). Suppose \( m + k \mapsto = n + k \mapsto \) [call this supposition (*) for reference]. Then

\[
\begin{align*}
(m + k)\mapsto &= m + k \mapsto \text{ [Def. of \( + \)]} \\
&= n + k \mapsto \text{ [By the supposition (*)]} \\
&= (n + k)\mapsto \text{ [Definition of \( + \)]}
\end{align*}
\]

Thus \( (m + k)\mapsto = (n + k)\mapsto \). Because predecessors are unique (Postulate 2) \( m + k = n + k \). Thus by the inductive hypothesis, \( m = n \).

Therefore, \( m + p = n + p \) implies \( m = n \) for all \( p \). Since this argument does not depend on any assumptions regarding \( m \) and \( n \), it is valid for all \( m \) and all \( n \). □

To prove that multiplication is commutative, we will need the following lemmas (analogous to Proposition 2 and Lemma 1).

**Lemma 2:** 0 is an “annihilator”

For any natural number \( n \), it is the case that \( 0 \cdot n = 0 = n \cdot 0 \).

**Proof:** The algorithm for multiplication directly specifies that \( n \cdot 0 = 0 \).

The proof that \( 0 \cdot n = n \) is by induction on \( n \).

**Basis** The goal is to show that \( 0 \cdot 0 = 0 \). But this is directly from the algorithm for multiplication.

**Inductive Hypothesis** Assume that \( 0 \cdot k = 0 \) for some \( k \in \mathbb{N} \).
**Inductive Step** The goal is to show that $0 \cdot k^\frown = 0$.

\[
\begin{align*}
0 \cdot k^\frown &= 0 + 0 \cdot k & [\text{Definition of } \cdot] \\
&= 0 + 0 & [\text{Inductive Hypothesis}] \\
&= 0 & [\text{Definition of } +]
\end{align*}
\]

□

**Lemma 3:** Successors migrate in products

For any natural numbers $m$ and $n$,

\[
m^\frown \cdot n = m \cdot n + n.
\]

**Proof:** The proof is by induction on $n$. Assume $m$ is fixed.

**Basis** The goal is to show that $m^\frown \cdot 0 = m \cdot 0 + 0$. Clearly $m^\frown \cdot 0 = 0 = 0 + 0 = m \cdot 0 + 0$ all follow from the definitions of $+$ and $\cdot$.

**Inductive Hypothesis** Suppose that $m^\frown \cdot k = m \cdot k + k$ for some $k \in \mathbb{N}$.

**Inductive Step** The goal is to show that $m^\frown \cdot k^\frown = m \cdot k^\frown + k^\frown$.

\[
\begin{align*}
m^\frown \cdot k^\frown &= m^\frown + m^\frown \cdot k & [\text{Exercise}] \\
&= m^\frown + (m \cdot k + k) & [\text{Exercise}] \\
&\vdots \\
&= (m \cdot k^\frown + k)^\frown & [\text{Exercise}] \\
&= m \cdot k^\frown + k^\frown & [\text{Exercise}]
\end{align*}
\]

Because the proof does not depend on assumptions about $m$, it is true for all $m$. □

Some of the other laws are left as exercises.

**Exercises**

16. Prove that $1$ is the identity for multiplication. That is, $1 \cdot m = m$ and $m = m \cdot 1$. 
17. Write out the entire proof of Lemma 3 providing the justifications for each line of the calculation in the inductive step.

18. Prove that multiplication distributes over addition \([m \cdot (n + p) = m \cdot n + m \cdot p]\) by induction on \(p\). You can use any of the lemmas and propositions we have already proved.

   (a) Prove the basis: \(m \cdot (n + 0) = m \cdot n + m \cdot 0\).

   (b) Write the inductive hypothesis.

   (c) Prove the inductive step: \(m \cdot (n + k) = m \cdot n + m \cdot k\)

19. Prove that multiplication is associative \([m \cdot (n \cdot p) = (m \cdot n) \cdot p]\) by induction on \(p\).

   (a) Prove the basis: \(m \cdot (n \cdot 0) = (m \cdot n) \cdot 0\).

   (b) Write the inductive hypothesis.

   (c) Prove the inductive step: \(m \cdot (n \cdot k) = (m \cdot n) \cdot k\). Hint: Use the Law of Distribution, which you just proved.

20. Prove that multiplication is commutative. Hint: Use the two Lemmas we proved right before these exercises.

21. Prove the Law of Positivity: if \(m + n = 0\), then \(n = 0\). [Hint: Use Case Distinction.]

22. Prove the Law of Integrality: if \(m \cdot n = 1\), then \(n = 1\). [Hint: Show that if \(n \neq 1\), then \(m \cdot n \neq 1\). By Case Distinction, \(n \neq 1\) means that either \(n = 0\) or \(n = k\) for some \(k\).

23. Prove the Law of No Zero Divisors. Hint: \(m \cdot n = m + m \cdot n\) by definition of multiplication. No explicit induction is needed for this proof.

For the record, we don’t prove the cancellation law for multiplication here. We return to it Chapter 11 when we have other techniques.
Examples of Recursion and Induction

Arithmetic operations on natural numbers like sum and product have been defined by recursion. In this chapter, we look at several other operations that arise in applications. Some of these, such as factorials, are familiar to you. Others are less so.

**Algorithm 3: Factorial**

For natural number $n$, define the *factorial of $n$, written $n!$, by the following:

\[
0! = 1 \\
(k\rightarrow)! = k! \cdot k\rightarrow
\]

Two closely related computations are knowns are rising and falling exponents.

**Algorithm 4: Rising exponent**

For natural numbers $n$ and $m$, define the *rising $n^{th}$ exponent of $m$, written $m^n$, by the following:

\[
m^0 = 1 \\
m^{k\rightarrow} = m \cdot (m^{k\rightarrow})^n
\]

We will also need falling exponents such as $5^\downarrow = 5 \cdot 4 \cdot 3$. Notice that $5^\downarrow$ should equal $5!$, but $5^{10}$ is not clear. For our present purpose, we...
will only define \( m^n \) when \( n \leq m \). This avoids the trouble of thinking about higher exponents, and is sufficient for our needs.

Recalling that \( n \leq m \) really means that \( d + n = m \) for some \( d \), we can define \( m^n \) by thinking of it as \( (d + n)^m \).

**Algorithm 5: Falling exponent**

For natural numbers \( d \) and \( n \), define the **falling exponent** \( (d + n)_n^m \) by recursion:

\[
(d + 0)_n^m = 1 \\
(d + k)^{k+1}_n^m = (d + k)^{k}_n^m \cdot (d + k)^k_n^m
\]

The sense in which rising exponent generalizes factorial is summarized in the following lemmas. The first two establish a law similar to the familiar fact about standard exponents that \( b^{x+y} = b^x \cdot b^y \).

**Lemma 4: Successor in rising exponents**

For any natural numbers \( m \) and \( n \),

\[ m^{n+1} = m^n \cdot (m + n) \]

**Proof:** The proof is by induction on \( n \).

**Basis** The goal is to show that \( m^{n+1} = m^n \cdot (m + 0) \) for all \( m \in \mathbb{N} \). But

\[ m^{n+1} = m \cdot (m^n) = m \]

And likewise \( m^n \cdot (m + 0) = m \).

**Inductive hypothesis** Suppose that for some \( k \in \mathbb{N} \), it is the case that

\[ m^{k+1} = m^k \cdot (m + k) \]

for all \( m : \mathbb{N} \).

**Inductive Step** The goal is to show that \( m^{k+2} = m^{k+1} \cdot (m + k) \) for all \( m \in \mathbb{N} \).

\[
\begin{align*}
    m^{k+2} &= m \cdot (m^k)^{k+2} \\
    &= m \cdot (m^k)^k \cdot (m^k + k) \\
    &= m^{k+1} \cdot (m + k^k)
\end{align*}
\]

\( \square \)
PROPOSITION 5: The law of rising exponents

For any natural numbers \( m, n \) and \( p \),
\[
m^{n+p} = m^n \cdot (m + n)^p.
\]

PROOF: The proof is by induction on \( p \).

**Basis** The goal is to show that \( m^{n+0} = m^n \cdot (m + n)^0 \). This is clearly true for any \( m \) and \( n \) because \( n + 0 = n \) and \( (m + n)^0 = 1 \).

**Inductive hypothesis** Suppose that for some \( k \in \mathbb{N} \),
\[
m^{n+k} = m^n \cdot (m + n)^k
\]
for all natural numbers \( m \) and \( n \).

**Inductive step** The goal is to show that
\[
m^{n+k+1} = m^n \cdot (m + n)^{k+1}
\]
for any \( m \) and \( n \).
\[
m^{n+k+1} = m^{n+k} \cdot (m + n)
= m^n \cdot (m + n)^k \cdot (m + n)
= m^n \cdot (m + n)^{k+1}
\quad \text{[By Lemma 4]}
\]
\[\Box\]

LEMMA 5: Factorial is a special case of rising exponent

For any natural number \( n \),
\[
n! = 1^n
\]

PROOF: We prove this by induction on \( n \).

**Basis** The goal is to show that \( 0! = 1^0 \). But \( 0! = 1 \) and \( 1^0 = 1 \) by definition.

**Inductive hypothesis** Suppose that \( k! = 1^k \) for some \( k \),
Inductive step  The goal is to show that \((k^\downarrow)! = 1^{k^\uparrow}\).

\[
1^{k^\uparrow} = 1^k \cdot (1 + k) \\
= k! \cdot k^\uparrow \\
= (k^\uparrow)!
\]

□

A law of falling exponents is analogous to the law of rising exponents.

PROPOSITION 6: The law of falling exponents

For any natural numbers \(m, n\) and \(p\),

\[
(m + n + p)^{n+p} = (m + n + p)^p \cdot (m + n)^n.
\]

PROOF: This is an exercise. □

Finally, rising and falling exponents are related by the following.

LEMMA 6: \((m^\uparrow)^{\downarrow} = (m + n)^\downarrow\)

PROOF: By induction on \(n\), we prove that for all \(m\), the lemma holds.

Basis: Both falling and rising exponents yield 1 for the exponent 0. So for all \(m\), \((m^\downarrow)^0 = (m + 0)^0\).

Inductive hypothesis: Suppose that for some \(k\), it is the case that \((m^\downarrow)^k = (m + k)^k\) holds for all \(m\).

Inductive step: The goal is to show that \((m^\downarrow)^{k+1} = (m + k^\uparrow)^{k+1}\) for all \(m\). Fix an arbitrary \(m\). Then

\[
(m^\downarrow)^{k+1} = (m + 1) \cdot ((m + 1)^\downarrow)^k \quad \text{[Definition]} \\
= (m + 1)^{\downarrow} \cdot (m + 1 + k)^k \quad \text{[Inductive hypothesis]} \\
= (m + 1 + k)^{1+k} \quad \text{[Law of falling exponents]} \\
= (m + k^\uparrow)^{k+1}
\]

□
4.1 Summation

You are probably familiar with notation like $\sum_{i=2}^{10} i^2$, meaning $2^2 + 3^2 + \cdots + 10^2$. We devote Chapter ?? to understanding how to calculate using $\sum$, but for now just look at basic definitions and a couple of examples.

In $\sum_{i=2}^{10} i^2$, 2 is called the lower limit of the sum and 10 is the upper limit. We can define $\sum$ recursively.

**Algorithm 6: Summation**

Suppose that $a_i$ is a real number for every $i$ between $m$ and $n$, including both. Then $\sum_{i=m}^{n} a_i = a_m + a_{m+1} + \ldots + a_n$. That is,

$$\sum_{i=m}^{m+k-1} a_i = 0$$

$$\sum_{i=m}^{m+k-1} a_i = (\sum_{i=m}^{m+k-1} m + k - 1a_i) + a_{m+k}$$

The base case may seem strange, but it makes sense if you consider that $\sum_{i=m}^{m} a_i$ ought to be the sum of a single term $a_m$. This suggests that $\sum_{i=m}^{m-1} a_i$ ought to equal 0.

Consider for an example, $\sum_{i=0}^{9} 2^i$. This is $2^0 + 2^1 + 2^2 + \cdots + 2^9$. We could simply evaluate this directly, but a more informative approach is to try to understand $\sum_{i=m}^{n} a_i$ for arbitrary $m \leq n$. The Table ?? shows the first several values of $\sum_{i=0}^{n} 2^i$ for small $n$, suggesting that the sum is always one less than a power of 2. In fact, $1 + \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all $n$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\sum_{i=0}^{n} 2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 + 2 = 3</td>
</tr>
<tr>
<td>2</td>
<td>1 + 2 + 4 = 7</td>
</tr>
<tr>
<td>3</td>
<td>1 + 2 + 4 + 8 = 15</td>
</tr>
<tr>
<td>4</td>
<td>1 + 2 + 4 + 8 + 16 = 31</td>
</tr>
</tbody>
</table>

Table 4.1: Sums of powers of 2

**Exercises**

24. Prove that $1 + \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

25. Prove that for any natural numbers $m \leq n \leq p$, and any summa-
tion $\sum_{i=m}^{p-1} a_i$, it is the case that

$$\sum_{i=m}^{p-1} = \sum_{i=m}^{n-1} a_i + \sum_{i=m}^{p-1} a_i.$$  

[Hint: This is a generalization of associativity of addition: $(a_0 + a_1) + (a_2 + a_3) = a_0 + a_1 + a_2 + a_3$. Your proof will use associativity at a key step.]

26. Prove that for any natural numbers $m \leq n$, any constant $c$ and any summation $\sum_{i=m}^{n} a_i$, it is the case that

$$\sum_{i=m}^{n} c \cdot a_i = c \sum_{i=m}^{n} a_i.$$  

[Hint: This is a generalization of distributivity: $c \cdot (a_0 + a_1) = c \cdot a_0 + c \cdot a_1$. Your proof will use distributivity at a key step.]

27. Prove that for any natural numbers $m \leq n$ and any summations $\sum_{i=m}^{n} a_i$ and $\sum_{i=m}^{n} b_i$, it is the case that

$$\sum_{i=m}^{n} a_i + b_i = \sum_{i=m}^{n} a_i + \sum_{i=m}^{n} b_i.$$  

[Meta-hint: This is a generalization of another familiar law of arithmetic.]

Remark: Recall from calculus that

$$\int_{m}^{n} f(x)dx = \int_{m}^{n} f(x)dx + \int_{n}^{p} f(x)dx,$$

$$\int_{m}^{n} cf(x)dx = c \int_{m}^{n} f(x)dx,$$

and

$$\int_{m}^{n} f(x) + g(x)dx = \int_{m}^{p} f(x)dx + \int_{m}^{p} g(x)dx.$$

These exercises point toward a valuable analogy between integration and summation. In Chapter ??, we explore this in depth.

Other ways to control indices

It is fairly common to see summations such as $\sum_{i=0}^{n} x^{n-i}i^i$. The exponent $n - i$ is related to $i$ and to the upper limit $n$ by the obvious fact that $i + (n - i) = n$. So we can write $\sum_{i+j=n} a_{i,j}$ as an abbreviation for the sum of all terms $a_{i,j}$ where $i + j = n$. We will look at generalizing
this to other conditions. For example, \( \sum_{i+j+k=n} a_{i,j,k} \) denotes the sum of all terms \( a_{i,j,k} \) where \( i + j + k = n \). For now, though, we only need the case of \( i + j = n \).

### 4.2 Binomial coefficients

A binomial is a term of the form \((x + y)^n\) for some natural number \( n \). Evidently \((x + y)^0 = 1\). And \((x + y)^k = (x + y) \cdot (x + y)^k = x(x + y)^k + y(x + y)^k\). Evaluating this exponent recursively, one sees that the result is a sum of terms \( x^iy^j \) where \( i + j = n \). That is,

\[
(x + y)^n = \sum_{i+j=n} \binom{n}{i} x^iy^j
\]

where \( \binom{n}{i} \) is some suitable choice of natural number coefficient. The goal of this section is to employ inductive reasoning to determine the values \( \binom{n}{i} \). This shows that the idea of inductive proof can be turned around to help us discover facts about natural numbers.

The basic case \((x + y)^0 = 1\) can be written as \( \sum_{i+j=0} x^iy^j \). The only values \( i \) and \( j \) satisfying \( i + j = 0 \) are \( i = 0 \) and \( j = 0 \). So apparently \( \binom{0}{0} = 1 \). We do not yet know any other values \( \binom{n}{i} \), so we must do a bit of detective work.

#### Exercises

28. Clearly, \((x + y)^0 = 1\), \((x + y)^1 = x + y\), and \((x + y)^2 = x^2 + 2xy + y^2\). Write the binomials \((x + y)^3\) and \((x + y)^4\) similarly.

Suppose we have already calculated values \( \binom{n}{i} \) for \( i \leq n \). So

\[
(x + y)^n = \sum_{i+j=n} \binom{n}{i} x^iy^j.
\]

We now look for values \( \binom{n}{i} \) for \( i \leq n \), so that

\[
(x + y)^n = \sum_{i+j=n} \binom{n}{i} x^iy^j.
\]

So start investigating the values \( \binom{n}{i} \), we can use the definition of
exponentiation:

\[(x + y)^n = (x + y) \cdot (x + y)^n = x \sum_{i+j=n} \binom{n}{i} x^i y^j + y \sum_{i+j=n} \binom{n}{i} x^i y^j = \sum_{k+j=n} \binom{n}{i} x^i y^j + \sum_{i+\ell=n} \binom{n}{k} x^i y^\ell,\]

where I have replaced the index \(i\) with \(k\) in the first summation and \(j\) with \(\ell\) in the second summation in order to keep the four indices distinct.

We still do not know much about the numbers \(\binom{n}{i}\), except that if we know the values \(\binom{n}{i}\) for \(i = 0\) through \(i = n\), then we should be able to infer the values of \(\binom{n}{i}\) for \(i = 0\) through \(i = n\).

The condition \(k + j = n\) falls into two cases. Either \(j = 0\) and \(k = n\), or \(j\) has a predecessor \(\ell\) and \(k + \ell = n\). Thus

\[\sum_{k+j=n} \binom{n}{k} x^k y^j = \binom{n}{0} x^n y^0 + \sum_{k+\ell=n} \binom{n}{k} x^k y^\ell.\]

Likewise, \(i + \ell = n\) falls into two cases. Either \(i = 0\) and \(\ell = n\), or \(i\) has a predecessor \(k\), and \(k + \ell = n\). So similarly,

\[\sum_{i+\ell=n} \binom{n}{i} x^i y^\ell = \left(\sum_{k+\ell=n} \binom{n}{k} x^k y^\ell\right) + \binom{n}{0} x^0 y^n.\]

Now notice that the conditions \(k + \ell = n\), \(k + \ell = n\), and \(k + \ell = n\) are all the same. So

\[(x + y)^n = \binom{n}{0} x^n y^0 + \left(\sum_{k+\ell=n} \binom{n}{k} x^k y^\ell\right) + \binom{n}{0} x^0 y^n.\]

Now back to thinking about \(\binom{n}{i}\). The summation that we are aiming for is \(\sum_{i+j=n} \cdots\). The indices \(i\) and \(j\) fall into three cases. Either \(i = 0\), or \(j = 0\), or neither. Hence,

\[(x + y)^n = \binom{n}{n} x^n y^0 + \left(\sum_{k+\ell=n} \binom{n}{k} x^k y^\ell\right) + \binom{n}{0} x^0 y^n.\]

Lining up the terms with like exponents for \(x\) and \(y\), we see that the
following must be true:

\[
\binom{n}{k} = \binom{n}{0}, \quad \binom{n}{k} = \binom{n}{n}, \quad \text{and} \quad \binom{n}{k} = \binom{n}{k} + \binom{n}{k}.
\]

We don’t quite have a proof of something yet. But we now have a very good idea of what must be true (and easily proved thanks to the work we’ve done). We simply need to translate these equations into a recursive definition of \(\binom{n}{i}\) in general, and then prove that

\[(x + y)^n = \sum_{i+j=n} \binom{n}{i} x^i y^j\]

following the same reasoning as the preceding paragraphs.

**Algorithm 7: Binomial coefficients**

For any two natural numbers, \(i \leq n\), we define \(\binom{n}{i}\) recursively.

To do that, recall that \(i \leq n\) means exactly that \(i + j = n\) for some \(j\). So we have an indirect way to define \(\binom{n}{i}\) by recursion on \(j\) and \(i\). That is, we define \(\binom{i+j}{i}\) explicitly. In the right column below, we also record the way \(\binom{n}{i}\) is usually stated (what you will find if look it up on your favorite search engine).

\[
\begin{align*}
\binom{0+0}{0} &= 1 & \binom{0}{0} &= 1 \\
\binom{0+\ell}{0} &= \binom{0+\ell}{0} & \binom{n}{0} &= 1 \\
\binom{k+0}{k} &= \binom{k+0}{k} & \binom{n}{n} &= 1 \\
\binom{k+\ell}{k} &= \binom{k+\ell}{k} + \binom{k+\ell}{k} & \binom{n}{k} &= \binom{n}{k} + \binom{n}{k}
\end{align*}
\]

for \(k < n\).
EXAMPLE 2:

To compute $5\text{choose}3$ by this algorithm, we first must decompose 5 as $2 + 3$. Then

$$\binom{3+2}{3} = \binom{3+1}{3} + \binom{2+2}{2}$$

$$= \binom{3+0}{3} + \binom{2+1}{2} + \binom{2+1}{2} + \binom{1+2}{1}$$

$$= 1 + \binom{2+0}{2} + \binom{1+1}{1}$$

$$+ \binom{2+0}{2} + \binom{1+1}{1} + \binom{1+1}{1} + \binom{0+2}{0}$$

$$= 1 + 1 + \binom{1+0}{1} + \binom{0+1}{0}$$

$$+ 1 + \binom{1+0}{1} + \binom{0+1}{0} + \binom{1+0}{1} + \binom{0+1}{0} + 1$$

$$= 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$$

$$= 10$$

LEMMA 7: The binomial coefficients satisfy $$(x + y)^n = \sum_{i+j=n} \binom{n}{i} x^i y^j$$

PROOF: Exercise □

Exercises

29. Calculate $\binom{4}{2}$ by explicit steps as in Example 2.

30. Calculate $\binom{5}{2}$ by explicit steps as in Example 2.


32. Prove that for every $j$ and $k$, $\binom{j+k}{j} = \binom{k+j}{k}$. Putting it in more traditional form, $\binom{n}{k} = \binom{n}{n-k}$ for any $k \leq n$. 
Clearly, based on your experience with these exercises, calculating binomial coefficients for large values by this algorithm is not practical. Another method of calculation is needed.

**Lemma 8:** For any natural numbers $n$ and $k$ where $k \leq n$, it is the case that \( \binom{n}{k} = \frac{n!}{k!(n-k)!} \).

We prove this in the form: for any natural numbers $i$ and $j$,

\[
i! \cdot j! \cdot \binom{i+j}{i} = (i+j)!
\]

**Proof:** We proceed by induction on $j$ to prove that the equation holds for all $i$.

**Basis:** For any $i$, \( \binom{i+0}{i} = 1 \). So $i! \cdot 0! \cdot \binom{i+0}{i} = (i+0)!$.

**Inductive hypothesis (1):** Suppose that for some fixed $\ell$, it is the case that

\[
i! \cdot \ell! \cdot \binom{i+\ell}{i} = (i+\ell)!
\]

holds for all $i$.

**Inductive step:** The goal is to show that

\[
i! \cdot \ell^{\ominus}! \cdot \binom{i+\ell^{\ominus}}{i} = (i+\ell^{\ominus})!
\]

holds for all $i$. For this, a proof by induction on $i$ will work.

**Basis:** $0! \cdot \ell^{\ominus}! \cdot \binom{0+\ell^{\ominus}}{0} = \ell^{\ominus}! = (0+\ell^{\ominus})!$.

**Inductive hypothesis (2):** Suppose it is the case that

\[
k! \cdot \ell^{\ominus}! \cdot \binom{k+\ell^{\ominus}}{k} = (k+\ell^{\ominus})!
\]

for some fixed $k$.

**Inductive step:** The goal of the inner inductive step is to show that

\[
k^{\ominus}! \cdot \ell^{\ominus}! \cdot \binom{k^{\ominus}+\ell^{\ominus}}{k^{\ominus}} = (k^{\ominus}+\ell^{\ominus})!.
\]
By a simple calculation,

\[ k^\cdot \ell^\cdot \binom{k^\cdot \ell}{k^\cdot \ell} = k^\cdot \ell^\cdot \binom{k^\cdot \ell}{k^\cdot \ell} + k^\cdot \ell^\cdot \binom{k^\cdot \ell}{k} \]

\[ = \ell^\cdot k^\cdot \ell^\cdot \binom{k^\cdot \ell}{k^\cdot \ell} + k^\cdot k^\cdot \ell^\cdot \binom{k^\cdot \ell}{k^\cdot \ell} \]

\[ = \ell^\cdot (k^\cdot \ell)! + k^\cdot k^\cdot \ell^\cdot \binom{k^\cdot \ell}{k^\cdot \ell} \]

\[ = (k^\cdot \ell)! + (k^\cdot \ell)! \]

Thus the inner inductive step is proved, showing that for all \( k \), it is the case that \( i^\cdot k^\cdot \ell^\cdot \binom{i^\cdot k}{k} = (i^\cdot k)! \). And since this is the goal of the outer inductive step, the lemma is proved.

\[ \square \]

The lemma makes calculations easier. For example, \( \binom{5}{3} = \frac{5!}{3! \cdot 2!} = \frac{120}{6} = 10 \) — supposing we know how to compute divisions, this is faster than directly using Algorithm 7.

Many alternative formulations of Lemma 8 can be derived by translating factorials as rising or falling exponents. For example, a special case of Lemma 5 is \( j^\cdot k^\cdot \ell^\cdot \binom{j^\cdot k}{k} = (j^\cdot k)! \). So

\[ k^\cdot \binom{j^\cdot k}{k} = (j^\cdot k)! \]

Exercises

33. Prove that \( \sum_{i=0}^{n} \binom{n}{i} = 2^n \) for every \( n \in \mathbb{N} \). [Hint: Consider what happens in \( (x + y)^n \) when \( x = y = 1 \).]

4.3 Fibonacci

The Fibonacci sequence is, historically, one of the first simple sequences that is defined by recursion. It shows up in many surprising places,
including analysis of some algorithms. There is even a data structure called a *Fibonacci heap*. We define the Fibonacci sequence as an operation on natural numbers, mainly for future use.

**Algorithm 8:**

For a natural number \( n \), define the natural number \( \text{fib}(n) \) by the following:

\[
\begin{align*}
\text{fib}(0) &= 0 \\
\text{fib}(1) &= 1 \\
\text{fib}(k \mapsto k) &= \text{fib}(k \mapsto k) + \text{fib}(k)
\end{align*}
\]

The first several values are presented in Table 4.2

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \text{fib}(n) )</th>
<th>( n )</th>
<th>( \text{fib}(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>55</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
<td>89</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>12</td>
<td>144</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>13</td>
<td>233</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>14</td>
<td>377</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>15</td>
<td>___</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>16</td>
<td>___</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>17</td>
<td>___</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>18</td>
<td>___</td>
</tr>
<tr>
<td>9</td>
<td>34</td>
<td>19</td>
<td>___</td>
</tr>
</tbody>
</table>

**Exercises**

34. Complete the remaining entries in Table ??.

35. Show that \( \text{fib}(n + 3) = 2 \cdot \text{fib}(n + 1) + 1 \cdot \text{fib}(n) \) for all \( n \).

36. Show that \( \text{fib}(n + 4) = 3 \cdot \text{fib}(n + 1) + 2 \cdot \text{fib}(n) \).

37. Show that \( \text{fib}((m + n) \mapsto) = \text{fib}(m \mapsto) \cdot \text{fib}(n \mapsto) + \text{fib}(m) \cdot \text{fib}(n) \) for all \( m \) and \( n \).
5

Ordering and Strong Induction

The standard ordering of natural numbers \( m \leq n \) does not need much explanation. But in fact, it arises in a precise way from addition. This fact permits us to reason about order of numbers using the arithmetic.

**Chapter Goals**

In this chapter, we define \( m \leq n \) and \( m < n \) for natural numbers and derive some useful properties of these relations. This leads to reformulations of induction that will be useful later.

**Definition 2: Comparison of Natural Numbers**

For natural numbers \( m \) and \( n \), say that \( m \) is less than or equal to \( n \) (written \( m \leq n \)) if and only if \( m + d = n \) for some \( d \in \mathbb{N} \). Also say that \( m \) is strictly less than \( n \) (\( m < n \)) if and only if \( m + d^\dagger = n \) for some natural number \( d \).

For example, \( 4 \leq 9 \) because \( 4 + 5 = 9 \). On the other hand \( 5 \nleq 3 \) because there is no natural number \( d \) for which \( 5 + d = 3 \). Likewise, \( 5 \nleq 5 \) because there is no \( d \) for which \( 5 + d^\dagger = 5 \).

From this definition, we can immediately infer some useful properties of \( \leq \).

**Reflexivity** For every \( m \), it is the case that \( m \leq m \). This is because \( m + 0 = m \). We say that \( \leq \) is reflexive.

**Transitivity** For every \( m \), \( n \) and \( p \), if \( m \leq n \) and \( n \leq p \), then \( m \leq p \).

For if \( m \leq n \), then \( m + d = n \) for some \( d \). Similarly, if \( n \leq p \), then \( n + e = p \) for some \( e \). So using associativity of addition, \( m + (d + e) = (m + d) + e = n + e = p \). We say that \( \leq \) is transitive.

**Anti-symmetry** For every \( m \) and \( n \), if \( m \leq n \) and \( n \leq m \), then \( m = n \).

For if \( m \leq n \) and \( n \leq m \), then \( m + d = n \) and \( n + e = m \) for some \( d \) and \( e \). So (using associativity) \( m + (d + e) = m + 0. \) But addition is
also cancellative. So \( d + e = 0 \). Now the Law of Positivity (22) tells us that \( d = 0 \). Hence \( m + 0 = n \).

We list. For example, \( m \leq n \) is always true because \( m + 0 = m \). If \( m \leq n \) and \( n \leq p \) then \( m \leq p \). And if \( m \leq n \) and \( n \leq m \), then \( m = n \).

I leave the proofs of these and a few other facts as exercises.

Another property of \( \leq \) that is a bit more subtle to prove is linearity. That is, for any two natural numbers \( m \) and \( n \), either \( m \leq n \) or \( n < m \).

**Lemma 9:** \( \leq \) is linear

For all \( m \in \mathbb{N} \) and \( n \in \mathbb{N} \), either \( m \leq n \) or \( n < m \).

**Proof:**

We proceed by induction on \( m \).

**Basis** Clearly, \( 0 \leq n \) for every \( n \) because \( 0 + n = n \).

**Inductive hypothesis** Suppose that for some \( k \in \mathbb{N} \), it is the case that for every \( n \in \mathbb{N} \), either \( k \leq n \) or \( n < k \).

**Inductive step** The goal is to show that for every \( n \in \mathbb{N} \), either \( k \leq n \) or \( n < k \).

Suppose \( n < k \). Then \( n + d = k \) for some natural number \( d \). So \( n + d = k \), and hence \( n < k \).

Suppose \( k \leq n \). Then \( k + d = n \) for some natural number \( d \). Now \( d \) must either be 0 or some successor. If \( d = 0 \), then \( k = n \). So \( n + 1 = n = k \). Hence \( n < k \). If \( d \) is a successor, then \( d = e \) for some \( e \). So \( k + e = k + e = n \). That is, \( k \leq n \).

Both cases have been accounted for, so either \( k \leq n \) or \( n < k \).

\[ \square \]

Linearity justifies the intuition that natural numbers are ordered so they can be arranged in a straight line with smaller numbers to the left and larger ones to the right. In other words, this jibes with the pictures we have been drawing all along.

Now that we know the natural numbers are linearly ordered, we can return to the question of cancellativity for multiplication — having punted on a proof of this earlier.
**Proposition 7:** Multiplication by a non-zero is cancellative.

For all natural numbers \(m, n\) and \(p\), if \(m \cdot p^\land = n \cdot p^\land\) then \(m = n\).

**Proof:** Suppose \(m \cdot p^\land = n \cdot p^\land\). Because of linear ordering of the natural numbers, either \(m \leq n\) or \(n < m\). Without loss of generality, assume \(m \leq n\). That is, \(m + d = n\) for some \(d\). So

\[
m \cdot p^\land + 0 = n \cdot p^\land = (m + d) \cdot p^\land = m \cdot p^\land + d \cdot p^\land.
\]

By cancellativity of addition, \(0 = d \cdot p^\land\). So by the Law of No Zero Divisors, \(d = 0\). \(\square\)

The ordering on natural numbers permits us to think about a new form of induction, sometimes called *strong induction* even though it is equivalent to simple arithmetic induction.

Suppose \(P(k)\) is a statement about natural numbers. That is \(P(5)\) might be true or false, \(P(2918)\) must be true or false, and so on. Let \(P^\ast(k)\) be the statement: \(P(j)\) is true for all \(j < k\). So \(P^\ast(5)\) is true just in case \(P(0), P(1), \ldots, P(4)\) are all true.

An inductive proof that \(P^\ast(m)\) holds for all \(m\) looks like this:

**Strong Induction, Preliminary Version**

Suppose \(P(m)\) is a property of natural numbers. Let \(P^\ast(m)\) be the related property so that \(P^\ast(m)\) holds if and only if \(P(j)\) holds for each \(j < m\). Then the following is an outline of a proof by induction that \(P^\ast(m)\) holds for all \(m\).

**Basis:** There is no \(j < 0\). So it is automatically true that \(P(j)\) holds for all \(j < 0\). In other words, \(P^\ast(0)\) holds no matter what \(P\) is.

**Inductive Hypothesis:** Suppose \(P^\ast(k)\) holds for some natural number \(k\). That is, \(P(j)\) holds for every \(j < k\).

**Inductive Step:** The goal is to prove that \(P^\ast(k^\land)\) holds. But by the inductive hypothesis, \(P(j)\) holds for every \(j < k\). So if \(P(k)\) also holds, then \(P^\ast(k^\land x t)\) holds. So the inductive step for \(P^\ast(k^\land)\) is complete just by proving that \(P(k)\) holds.

We can simplify this into two parts (because the basis is trivial).
**Strong Induction, usable version**

**Strong inductive hypothesis:** Suppose that for some \( k \), it is the case that \( P(j) \) holds for all \( j < k \).

**Strong inductive step:** Prove that \( P(k) \) holds.

Conclude by strong induction that \( P(m) \) holds for all \( m \).

---

**Example 3:**

We claim that the Fibonacci numbers grow exponentially fast. In particular, \( \phi^n \leq \text{fib}(n+2) \), where \( \phi = \frac{1 + \sqrt{5}}{2} \). First, check for yourself that \( \phi^2 = \phi + 1 \).

The proof is by strong induction.

**Strong Inductive Hypothesis** Suppose that for some \( k \), it is the case that \( \phi^j \leq \text{fib}(j+2) \) holds for all \( j < k \).

**Inductive Step** For \( k < 2 \), check that \( \phi^0 = 1 \leq \text{fib}(2) \) and \( \phi^1 = \phi \leq \text{fib}(3) \). So the claim is true for \( k = 0 \) and \( k = 1 \). For values \( k \geq 2 \), \( k \) as \( j \rightarrow \) for some \( j \). So

\[
\phi^k = \phi^2 \cdot \phi^j = (\phi + 1)\phi^j = \phi^{j+1} + \phi^j.
\]

By the strong inductive hypothesis, \( \phi^j \leq \text{fib}(k-j) \) and \( \phi^j \leq \text{fib}(k) \). So \( \phi^k \leq \text{fib}(k+2) \).

Hence, \( \phi^n \leq \text{fib}(n+2) \) for all \( n \).

---

**Minimization**

Suppose \( P(\cdot) \) is a property of natural numbers. Let’s refer to any \( n \) for which \( P(n) \) holds as a **satisfier of** \( P \). A **least satisfier** of \( P \) is any \( n \) so that (i) \( n \) is a satisfier of \( P \) and (ii) if \( m \) is a satisfier of \( P \), then \( n \leq m \).

If \( P \) has any satisfier at all, it must have a least satisfier. For suppose \( P \) has no least satisfier. Then consider the property \( \neg P \) defined by \( \neg P(n) \) is true if and only if \( P(n) \) is false. Evidently, if \( \neg P(j) \) is true for every \( j < k \), then \( \neg P(k) \) must also hold. Otherwise, \( P(k) \) would hold, so \( k \) would be a satisfier, and since by the strong inductive hypothesis, no natural number less that \( k \) is a satisfier, \( k \) would have to be the
least satisfier. We supposed that no such thing exists. So by strong
induction, \( \neg P(k) \) holds for every \( k \). In other words, \( P(k) \) is false for all
\( k \).

We can restate this simply.

---

**THEOREM 1: Minimization Principle for Natural Numbers**

Any property of natural numbers that is satisfied by at least one
natural number has a least satisfier.

**PROOF:** The proof is covered essentially in the foregoing paragraph. \( \square \)

We will not illustrate this principle here, but you will encounter the
idea in other courses. It is sometimes a convenient way to think about
an inductive proof.

---

**5.1 Laws of ordered arithmetic**

The arithmetic laws for addition and multiplication, as stated in
Chapter ??, are all to do with equality. But now that we have the
ordering of natural numbers, we ought to spend at how \( \leq \) interacts
with arithmetic.

---

**LEMMA 10: Addition is monotonic**

For any natural numbers, \( m, n \) and \( p \), if \( m \leq n \), then \( m + p \leq n + p \).

**PROOF:** This is an exercise. \( \square \)

---

**LEMMA 11: Addition is order-cancellative**

For any natural numbers \( m, n \) and \( p \), if \( m + p \leq n + p \), then \( m \leq n \).

**PROOF:** This is an exercise. \( \square \)
LEMMA 12: Multiplication is monotonic

For any natural numbers \( m, n \) and \( p \), if \( m \leq n \), then \( m \cdot p \leq n \cdot p \).

PROOF: This is an exercise. □

LEMMA 13: Multiplication by a positive natural number is order-cancellicative

For any natural numbers \( m, n \) and \( p \), if \( m \cdot p \leq n \cdot p \), then \( m \leq n \).

PROOF: This is an exercise. □

Exercises

38. Prove Lemma 10.
40. Prove Lemma 12.

5.2 Monus: Subtraction in \( \mathbb{N} \)

We close this chapter by considering how to make sense of subtraction on natural numbers, where for example, \( 7 - 10 \) does not work. The idea is pretty simple. The main value in looking at it now is to provide an example of a general phenomenon that has more useful applications later.

If we were looking at integers, subtraction would be the usual inverse of addition. Subtraction of integers is related to addition by the fact that

\[
a = b + c \quad \text{if and only if} \quad a - b = c
\]

for any integers \( a, b \) and \( c \). In fact, this equivalence is precisely what defines subtraction for integers. But for natural numbers, for example, \( 13 - 7 \) makes sense, but \( 7 - 13 \) does not. On the other hand, we can relax the requirement a bit, and ask for an operation \( \div \) on natural
numbers (known by the obscure term \textbf{monus}) so that for all natural numbers \(m, n\) and \(p\),

\[ m \leq n + p \quad \text{if and only if} \quad m - n \leq p. \]

Suppose that \(n \leq m\). Then \(n + d = m\) for some \(d\). So \(m \leq n + p\) implies \(n + d \leq n + p\). Since addition is order cancellative, \(d \leq p\). Conversely, if \(d \leq p\), then \(m \leq n + p\) because addition is monotonic. So in the case that \(n \leq m\), the value of \(m - n\) is just the usual difference. For example \(13 - 5 = 8\).

Suppose \(m < n\). Then \(m \leq n + p\) is always true, no matter what \(p\) happens to be. And \(0 \leq p\) no matter what \(p\) happens to be. So \(m - n = 0\). Hence \(\cdot\) yields the usual subtraction when it can, and yields 0 when that is the best it can do.

An explicit algorithm for computing \(m - n\) is simple enough, but not very useful. We write it here just for the record.

\begin{algorithm}
\textsc{Algorithm 9: Monus}

For natural numbers \(m\) and \(n\), define \(m - n\) by the following recursion for any \(j\) and \(k\),

\[\begin{align*}
0 \cdot k &= 0 \\
j \cdot 0 &= j \\
j \cdot k &= j \cdot k
\end{align*}\]

The main point is that \(m - n\) always yields a natural number that is the “honest” subtraction when \(n \leq m\), and is 0 otherwise.

Clearly, \(m - n \leq m - n\). So \(m \leq n + (m - n)\) holds for any \(m\) and \(n\). Similarly, \(m + n \leq m + n\). So \((m + n) - m \leq n\). In fact, since \(m \leq m + n\), the result of \((m + n) - m\) is just the standard difference, so \((m + n) - m = n\).

\end{algorithm}

Exercises

42. Prove that \(\cdot\) is monotonic in its first argument. That is, if \(m \leq m'\), then \(m - n \leq m' - n\).

43. Prove that \(\cdot\) is antitonic in its second argument. That is, if \(n \leq n'\), then \(m - n' \leq m - n\).
44. Determine whether $\dot{\cdot}$ is order right cancellative, and prove your result. That is, is it the case that $m \dot{\cdot} n \leq m' \dot{\cdot} n$ implies $m \leq m'$?

45. Prove that $m \cdot (n \dot{\cdot} p) = (m \cdot n) \dot{\cdot} (m \cdot p)$.

46. Prove that $(m \dot{\cdot} n) \dot{\cdot} p = m \dot{\cdot} (n + p)$.

47. Show that $m + (n \dot{\cdot} p)$ is not necessarily equal to $(m + n) \dot{\cdot} p$. 
Part II

Sets, Functions, Relations and Proofs
THE MATHEMATICAL UNIVERSE consists of various types of mathematical objects: numbers, functions, graphs, lists, groups, vector spaces, linear operators, and so on. We need ways to talk about these things, in isolation and in relation to one another.

Suppose you are asked to solve the equation

\[ 0 = 2x^3 - x^2 + 6x - 3 \]

for \( x \). You would have a right to ask whether \( x \) is meant to be an integer (in which case there are no solutions), a real number (in which case there is one solution) or a complex number (in which case there are three solutions). So “solve for \( x \)” does not really mean anything until you know what type of \( x \) is being sought. This is no different than in ordinary conversation. You can not answer “What would you like?” unless you know what the question is about. Are you being asked what to have for dinner, what to get you for your birthday, what to do on vacation, or something else?

The development of programming languages has made clear the importance of data types. In most contemporary programming languages, each datum has an associated type. For example, a program might involve integer data separate from character data and separate from, say, floating point data (roughly, floating point data approximates real number data in a way that is computationally tractible). Other data types might include things like matrices, arrays, lists, and much more.

In mathematics, natural numbers, real numbers, integer polynomials, complex matrices, continuous functions on the reals and so on are essentially the mathematical counterparts of data types. They help us to organize the things in much the same way that data types do in java or C++.

Though mathematicians tend to use the idea of types more informally than do computer scientists, one of the important lessons learned from computer science is that closer attention to type information helps clarify how things are related.

A set is, in the formulation of Cantor, jedes Viele, welches sich als Eines denken lasst “any multiplicity which can be comprehended as one”. For example, several playing cards taken together form a single deck of cards; the deck is a multiplicity of cards comprehended as one thing. The several students taking Discrete Math right now can be comprehended as one class. The infinitely many natural numbers can be regarded as single thing, the set of natural numbers. Thus a set is essentially a collection of elements.

In a more “typed” way of thinking, the natural numbers constitute a type of mathematical data, distinct from, say, the type of real numbers, the type of \( n \times n \) matrices, the type of Python programs, etc. The important point is, for example, that you know what it means to add two natural numbers, or prove something about them by simple arithmetic induction. On the other hand, adding two Python programs makes no sense. Proving something about real numbers by simple arithmetic induction makes no sense (but other “inductive” techniques do). A “typed” way of thinking, the emphasis is put on what you can do to a datum. A type is essentially a specification of how to build data, what can be done to data of that type, and how to tell when data are equal. We hinted at this in Part I, when we developed the vocabulary and postulates for natural numbers. In effect, we presented the natural numbers as a type.

A function is a correlation of the members of one set with members of another set. Functions can be thought of and used in many, many ways. Here are some examples.

- Polynomials such as \( f(x) = x^3 + 2x^2 - x + 1 \) are functions. They can be used in a wide variety of modeling problems in their own right, but also as approximations of more complicated phenomena.
• The process of definite integration takes a given function \( f \) and produces a second function \( \int_0^x f(x) \, dx \). Integration is itself a “higher-order” function. Because it transforms one function into another.

• Cryptography is based almost entirely on the problem of designing functions with special properties. A crypto-system is based on two functions \( E \) (standing for “encrypt”) and \( D \) (“decrypt”). The function \( E \) will take a message \( m \) and an encryption key \( k_e \), and produce an encrypted message \( E(m, k_e) \). The function \( D \) will take an encrypted message \( c \) and a decryption key \( k_d \) and produce a clear message \( D(c, k_d) \). The system of these two functions is correct if it is the case that whenever \( k_e \) and \( k_d \) are correctly paired, \( D(E(m, k_e), k_d) = m \). In words, encrypting a message and then decrypting it with the matched key restores the original message. A correct system may still not be safe to use. To be minimally cryptographically safe, it must also be difficult computationally to determine \( m \) from \( E(m, k_e) \) and \( k_e \), and most also be difficult to determine \( k_d \) from \( k_e \).

• In programming, it is possible to implement a given process in many different ways. For example, one programmer may use a \texttt{while} loop, whereas another uses recursion. To understand how to compare two implementations, one needs to know that they both may implement the same function. In fact, software engineers refer to “functional specifications” when they consider this.

A relation is more or less what is sounds like. For example, \( \leq \) is a relation on natural numbers. It is either true (\( 4 \leq 7 \)) or not (\( 4 \not\leq 2 \)) for any two natural numbers. Relations do not have to be defined on data of one type. For example, “root of” might be a relation between polynomials and real numbers: \( 2 \) is a root of \( x^2 + x - 6 \), but \( 4 \) is not a root of \( x^2 + x - 6 \). In everyday life, we commonly think in terms of relations such as “Sam is a friend of Frodo”, “Frodo is shorter than Gandalf”, and so on.

Sets, functions and relations form an informal type-oriented framework in which virtually all of mathematics can be built. So an understanding of sets, functions and relations is key to a rigorous approach to most other parts of mathematics as well as of computing.

Sets, functions and relations are fundamental to modern mathematics and computation. Sets are comprehensible collections. Functions are a way of thinking about attributes of the things in a set, like “the color of”, “the mass of”, “the location of”, “the velocity of”, “the father of”, “the favorite book of the person to the left of” and so on. Relations are a formal way of thinking about how things can be related. For example, “less than” is a relation between numbers: \( 5 \) is less than \( 7 \), but \( 3 \) is not less than \( 2 \).

Set theory is roughly analogous in mathematics to a low level programming language in computer science. It provides a system in which we can describe and reason about a remarkably wide range of complicated structures. In programming, one might think about data types, programs and data base queries. In mathematics, the approximate analogues of these are sets, functions and relations. Though not perfect, this analogy is worth keeping in mind as you are learning to think about set theory.

Traditional set theory emphasizes the role that sets play, deriving the concepts of functions and relations from that. In fact, it is possible to develop a version of set theory that emphasizes functions, or one that emphasizes relations. But in practice, mathematicians usually think more directly in terms of interplay between the three. That will be our approach.

Sets, functions and relations are useful partly because they can model non-mathematical, real-world
phenomena. A simple example is a set modelling a standard poker deck. We might denote it by

\[
\text{DECK} = \{A\spadesuit, 2\spadesuit, 3\spadesuit, 4\spadesuit, 5\spadesuit, 6\spadesuit, 7\spadesuit, 8\spadesuit, 9\spadesuit, 10\spadesuit, J\spadesuit, Q\spadesuit, K\spadesuit, \\
A\heartsuit, 2\heartsuit, 3\heartsuit, 4\heartsuit, 5\heartsuit, 6\heartsuit, 7\heartsuit, 8\heartsuit, 9\heartsuit, 10\heartsuit, J\heartsuit, Q\heartsuit, K\heartsuit, \\
A\diamondsuit, 2\diamondsuit, 3\diamondsuit, 4\diamondsuit, 5\diamondsuit, 6\diamondsuit, 7\diamondsuit, 8\diamondsuit, 9\diamondsuit, 10\diamondsuit, J\diamondsuit, Q\diamondsuit, K\diamondsuit, \\
A\clubsuit, 2\clubsuit, 3\clubsuit, 4\clubsuit, 5\clubsuit, 6\clubsuit, 7\clubsuit, 8\clubsuit, 9\clubsuit, 10\clubsuit, J\clubsuit, Q\clubsuit, K\clubsuit\}
\]

The elements are arranged here conveniently, but we could just as well have listed the cards in any shuffled order. The set of them (that is, the deck) would be the same. The notation used braces \{ \ldots \} is a standard way to indicate that this is meant to be a set, So order and repetition are not important.

For any card in a deck, “the rank of” or “the suit of” are two attributes of each card, writing \(\text{rank}(A\spadesuit) = A\) and \(\text{suit}(A\spadesuit) = \spadesuit\). In general, \(\text{rank}(c)\) and \(\text{suit}(c)\) pick out the relevant attributes of any card \(c\). The potential values of these attributes also form sets

\[
\text{RANK} = \{A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\}
\]

and

\[
\text{SUIT} = \{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}.
\]

Formally, we model rank and suit as functions.

In a deck a five card hand is a subset of \text{DECK} consisting of five cards. The collection of all possible hands is sometimes denoted by \text{DECK}[^5]. We know that a four-of-a-kind hand beats a full-house. In general, there is a relation beats on hands to that \(h_1\) beats \(h_2\) holds if \(h_1\) is the winning hand between them. The relation beats can be broken down farther by establishing an order on ranks and suits (for example, \(\heartsuit\) is higher than \(\spadesuit\)), and an order on kinds of hands (any full house beats any pair, etc.)

The functions rank and suit capture the structure of the elements of \text{DECK}. The relation beats captures something about the structure of poker games. A slogan to remember is

**Functions and relations explain structure.**

To understand sets, functions and relations as they are used in every day mathematics, we need to answer some questions:

- What do we mean by saying that a set is a collection?
- What do we mean by saying that two sets are equal?
- What do we mean by saying that a function is an attribute?
- What do we mean by saying that two functions are equal?
- What do we mean by a relation?
- What do we mean by saying two relations are equal?
- How do we construct sets, functions and relations?

The last of these questions can be very roughly paraphrased to ask, what is the ‘programming language’ for sets? While there is no fixed answer to that (as there is not just one programming language), we will settle on a simple set of principles by which we can build almost everything a mathematician needs.
We want maximum flexibility for defining new sets, functions and relations. That suggests we might propose a fairly complicated language that supports as wide a variety of constructions as possible. But it turns out that is not necessary. Starting with a rather small menu of constructions, we can combine them to build most of the structures one needs for contemporary mathematics.
6

Sets, Functions, Relations

The fundamental building blocks of contemporary mathematics are sets, functions and relations. To understand mathematical reasoning (and its close cousin, computation), we need to understand how these interact. We have already encountered examples of all three notions. The collections \( \mathbb{N} \) and \( \mathbb{N}^+ \) are sets; operations such as addition and multiplication are functions; and \( \leq \) and \( < \) are relations. Here is a rough idea of what we mean.

- Sets correspond to types of mathematical data. For example, natural numbers form a set; real numbers form a set; real number matrices form a set; complex polynomials form a set.

- Functions correspond to data transformations. For example, the functions \( \sin \) and \( \cos \) transform an angle \( \theta \) into a corresponding vertical and horizontal displacements; the function \( \text{min} \) transforms a pair of natural numbers into the smaller of the two; the function \( + \) transforms a pair of natural numbers into their sum.

- Relations permit us to speak precisely about how data of one type may be related to data of another. For example, “less than” is a relation between natural numbers and natural numbers; “solves” is a relation between, say, real numbers and real number polynomial equations (as in “3 solves the equation \( 0 = x^2 - x - 6 \)”; “overlaps” is a relation between discs in the plane (some discs overlap each other, others do not).

The development of programming languages has highlighted the importance of data types. In most contemporary programming languages, each datum has an associated type. For example, a program might involve integer data separate from character data and separate from floating point data. Other computational data types include...
things like matrices, arrays, lists, trees and much more. But the typical programming language notion of data type is too restrictive for all mathematical needs. In computing, we are constrained in what can count as a data type because a program has to be executable on an actual computer. This is a serious limitation. We would not be able to deal properly with all real numbers, for example.

To avoid confusing computational data types from a more general concept, we will use the terms that mathematicians use.

6.1 Sets

A set is, in the formulation of Cantor, *jedes Viele, welches sich als Eines denken lasst* “any multiplicity which can be comprehended as one”. Cantor was getting at the idea that, for example, the collection of natural numbers $\mathbb{N}$ can be understood as a single entity, just as several playing cards taken together an be understood to be a single deck of cards. When a collection of individuals can be understood as a single entity in its own right, it is a set.

In Chapter 1, we introduced the notation $n \in \mathbb{N}$ and $n \in \mathbb{N}^+$ to indicate that $n$ is a natural number, or that $n$ is a positive natural numbers. We extend that notation to all sets.

**Vocabulary 2: Basic Vocabulary of Sets**

A set is an entity $A$ that we think of as consisting of elements.

We write $x \in A$ to indicate that $x$ is an element of $A$. Sometimes, we may also write $x \not\in A$ to indicate that $x$ is not an element of $A$. When $x \in A$, we say that $x$ is in $A$.

For variety, all of the following phrases mean the same thing:

- $x$ is in $A$
- $x$ is an element of $A$
- $x$ is a member of $A$
- $A$ contains $x$
- $x$ belongs to $A$

This vocabulary comes with some obligations. To describe a set, we have to provide a precise criterion for membership — that is, an unambiguous way to determine what is in and what is not in the set.
A collection for which the criterion of membership is vague is not a set. For example, “aging professors” does not constitute a set because “aging” is a vague term. On the other hand, “professors at least 45 years old” does constitute a set (assuming that precise age can be pinned down and assuming that professors are mathematical objects).

To describe a small set, we can simply list the elements, using braces ( { and } ) to indicate that we are describing a set as in the following example.

**Example 4:**

A standard deck of poker cards can be described by

\[
\text{DECK} = \{A\spadesuit, 2\spadesuit, 3\spadesuit, 4\spadesuit, 5\spadesuit, 6\spadesuit, 7\spadesuit, 8\spadesuit, 9\spadesuit, 10\spadesuit, J\spadesuit, Q\spadesuit, K\spadesuit, \\
A\heartsuit, 2\heartsuit, 3\heartsuit, 4\heartsuit, 5\heartsuit, 6\heartsuit, 7\heartsuit, 8\heartsuit, 9\heartsuit, 10\heartsuit, J\heartsuit, Q\heartsuit, K\heartsuit, \\
A\diamondsuit, 2\diamondsuit, 3\diamondsuit, 4\diamondsuit, 5\diamondsuit, 6\diamondsuit, 7\diamondsuit, 8\diamondsuit, 9\diamondsuit, 10\diamondsuit, J\diamondsuit, Q\diamondsuit, K\diamondsuit\}
\]

The elements are arranged here conveniently, but we could just as well have listed the cards in any shuffled order. The set — DECK itself — is the same regardless of how it is shuffled.

We can also describe a set of ranks and a set of suits:

\[
\text{RANK} = \{A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K\}
\]

\[
\text{SUIT} = \{\spadesuit, \heartsuit, \diamondsuit, \clubsuit\}.
\]

As you know, the deck consists of 52 cards: one card for each possible rank and suit. So DECK is, in a sense, built from RANK and SUIT by systematically pairing each rank with each suit. We will return the general process of building a set by pairing elements of two sets below.

**Set comparison**

Compare the sets \{1, 2, 3, 4, 5\} and \{1, 5, 4, 2, 3\}. Are they distinct or not? Remember, the idea is that a set is just a collection that can be precisely defined. So elements are either in or out. Nothing more. By that view, these two sets have exactly the same elements. So it makes sense to say they are equal. Likewise, the set DECK that we defined earlier is the same set when we shuffle the cards.

Compare the sets \{1, 3, 5, 7, 9\} and \{n ∈ \mathbb{N} \mid n \text{ is odd and } n < 10\}. 
Again, they are equal because every odd natural number less than 10 is in the set \{1, 3, 5, 7, 9\}, and vice versa.

This leads to a useful definition and a principle allowing us to determine when sets are equal.

**Definition 3: Subsets**

For sets \(X\) and \(Y\), we say that \(X\) is a subset of \(Y\) provided that every element of \(X\) is an element of \(Y\). We write this as \(X \subseteq Y\), and say that \(X\) is included in \(Y\), or \(Y\) includes \(X\). We may also write \(Y \supseteq X\) to mean the same thing, and say that \(Y\) is a superset of \(X\).

If \(X\) is not a subset of \(Y\), we write \(X \nsubseteq Y\). If \(X \subseteq Y\) and \(Y \nsubseteq X\), then \(X\) is called a proper subset of \(Y\). To indicate that \(X\) is a proper subset of \(Y\), we may write \(X \subset Y\). Some people write \(X \subsetneq Y\) for proper subsets, but I will never use that notation in this course.

To say \(X \subseteq Y\) is to say that for any \(x\), if \(x \in X\) then \(x \in Y\). In plain English, we may translate it as “all Xs are Ys.” Suppose \(P\) is the set of all professors, and \(H\) is the set of all human beings. Then \(P \subseteq H\) is the (dubious) assertion that “all professors are human beings.”

**Example 5:**

Here are some examples and counter-examples of the subset relation.

- \(\{1, 2, 3\} \subseteq \{0, 1, 2, 3\}\).
- \(\{1\} \nsubseteq \{0, 1\}\) because the only element of the first set is \(1\), and \(1 \notin \{0, 1\}\).
- \(X \subseteq X\) for any set \(X\) because, trivially, every element of \(X\) is an element of \(X\).
- \(\emptyset \subseteq X\) for any set \(X\) because every element of \(\emptyset\) (there are none) is an element of \(X\).
- \(\{1, 2, 3\} \nsubseteq \{0, 2, 3\}\) because \(1 \in \{1, 2, 3\}\) but \(1 \notin \{0, 2, 3\}\).
- \(\{1, 2, 3\} \subseteq \{2, 3, 1\}\).
- \(\{\spadesuit\} \subseteq \text{SUITS}\).

**Exercises**
For each of the following pairs of sets, determine whether or not the first is a subset of the second. Explain each answer in one sentence.

48. \{0, 1\} and \{1, 0\}
49. \{a, b, c, d\} and \{a, b, d, e, c\}
50. \{\} and \{\{\}\}
51. \{0, 3, 6, 10\} and \{10, 9, 8, 7, 6, 5, 4, 2, 1, 0\}
52. \{1, 2, 1\} and \{1, 2, 3\}

To say that two sets are equal is to say they have exactly the same elements. This leads us to the following principle.

**Principle 1: Set extensionality**

For sets \(X\) and \(Y\), if \(X \subseteq Y\) and \(Y \subseteq X\), then \(X = Y\).

**Example 6:**

Let \(A\) be the set of all prime numbers that are greater than 2 less than 15. Let \(B\) be the set of all odd numbers greater than 2 and less than 15, except 9. So 3 \(\in A\) because 3 is greater than 2, less than 15 and prime. Also 3 \(\in B\) because 3 is greater than 2, less than 15, odd and not equal to 9. It is easy to check that every element of \(A\) is odd and not equal to 9. So \(A \subseteq B\). Conversely, every element of \(B\) is prime. So \(B \subseteq A\). So \(A = B\).

**Basic set constructions**

To describe larger sets (including infinite ones) we can not list the elements. The most common way to specify a large set is to give a precise criterion for membership. The most common notation for this is illustrated in

\[ \{x \in \mathbb{N} \mid x \text{ is a perfect square}\} \]

A **perfect square** is a natural number that happens to be a square of a natural number. So 0, 1, 4, and 9 are perfect squares; 2, 3, 5 and
are not. So the same set could also be described informally as 
\[\{0, 1, 4, 9, \ldots\}\].

For a set \(X\), generally we can define subset of \(X\) by 
\[\{x \in X \mid \ldots\}\].

Whatever we write in place of \(\ldots\) must be a clear description of some property that elements of \(X\) may have. This method of definition is called set selection because it defines a subset of \(X\) by selecting elements to retain.

Some sets are familiar, and obviously useful to have around. We will not try to define them formally for now, but just give them names and informal descriptions.
**Some Important Sets**

The following sets are denoted by the special symbols:

\[
\begin{align*}
\mathbb{N} &= \text{the set of natural numbers: } 0, 1, 2, \ldots \\
\mathbb{N}^+ &= \text{the set of positive natural numbers: } 1, 2, 3, \ldots \\
\mathbb{Z} &= \text{the set of integers: } \ldots, -2, -1, 0, 1, 2, \ldots \\
\mathbb{Q} &= \text{the set of rational numbers: } \frac{1}{2}, \frac{24}{23} \text{ and so on} \\
\mathbb{R} &= \text{the set of real numbers} \\
\mathbb{C} &= \text{the set of complex numbers} \\
\emptyset &= \text{the empty set, having no elements}
\end{align*}
\]

These symbols are very common. You will be understood by any mathematician if you use them.

The set Deck is built by pairing each rank with each suit. Similarly, you are quite used to thinking of the 2 dimensional plane as consisting of “points” \((x, y)\) where \(x\) and \(y\) are both real numbers. So the plane can be “built” by pairing each real number (as an \(x\) coordinate) with each real number (as a \(y\) coordinate).

In general, we allow for pairing of any two sets in this way. The real utility of doing this will be clearer in Chapter 7.

**Vocabulary 3: Products of sets**

For any two sets \(X\) and \(Y\), there is a set \(X \times Y\) so that \(X \times Y\) consists of all pairs \((a, b)\) where \(a \in X\) and \(b \in Y\).

There is a set \(1\) that consists of exactly one element. For our purposes, we will write \(1 = \{\bullet\}\).

By this principle, we can think of the plane explicitly as \(\mathbb{R} \times \mathbb{R}\) (the abbreviation \(\mathbb{R}^2\) is also used). Also, we do not need to define an explicit set Deck by listing all 52 cards. We might as well define Deck = Rank \(\times\) Suit, so that a typical card actually is a pair such as \((5, \bullet)\).

The handy notation for the plane, \(\mathbb{R}^2\), generalizes to repeated products of any fixed set.
DEFINITION 4: Finite powers of a set

For a set $X$, each $n \in \mathbb{N}$ determines a finite power of $X$ as follows.

$$
X^0 = 1 \\
X^1 = X \\
X^k = X \times X^k \\
$$

for positive $k$.

This definition allows us to say that the two dimensional plane is $\mathbb{R}^2$, three dimensional space is $\mathbb{R}^3 = \mathbb{R} \times (\mathbb{R} \times \mathbb{R})$, and so on.

Exercises

53. Explain why $\mathbb{R}^2 \times \mathbb{R} \neq \mathbb{R}^3$.

6.2 Functions

Each card in the standard deck has a rank and a suit: the rank of $5\Diamond$ is 5, the suit of $Q\spadesuit$ is $\spadesuit$, and so on. So rank is an attribute of a card, and suit as another attribute. Put another way, rank and suit transform a card $c \in \text{DECK}$ into a rank, rank$(c) \in \text{RANK}$, or a suit, suit$(c) \in \text{SUIT}$.

VOCABULARY 4: Basic Vocabulary of Functions

For a set $X$ and a set $Y$, a function from $X$ to $Y$ is an entity $f$ with the following feature.

- Each element $a \in X$ determines an element $f(a) \in Y$, spoken as “$f$ of $a$”, or sometimes as “$f$ evaluated at $a$.”

  A function from $X$ to $X$ is also called a function on $X$. We also use the following notation and terms:

- $f: X \to Y$ indicates that $f$ is a function from $X$ to $Y$.

- For a function $f: X \to Y$, the set $X$ is called the domain of $f$ and $Y$ is called the codomain of $f$. The parentheses are sometimes omitted, writing $fa$ instead of $f(a)$. Also, for some special purposes, $f_a$ is an alternative notation. The point is that a function $f$ from $X$ to $Y$ is an entity in its own right, which together with an element $a \in X$ determines an element of $Y$. 

It can be helpful to picture a function as a machine that transforms input to output. We can depict $f: X \rightarrow Y$ as in Figure 6.1.

The trapezoid shape is not important. I only use it to suggest a direction from input (the big end) to output (the small end). Wiring diagrams are common in circuit design, where different shapes are frequently used to indicate different functions. We will investigate these sorts of diagrams (called wiring diagrams) in general in Chapter 7, and in the special case of electronic circuits in Chapter ??.

It can also be helpful to picture a function more abstractly as an arrow between sets as in Figure 6.2. A diagram like this is called an external diagram. We will also investigate external diagrams in Chapter 7.

Wiring diagrams and external diagrams convey similar information, but serve different purposes. Wiring diagrams are useful when we need to think about details of how functions are built up from other functions. External diagrams are useful when we need to think about how functions interact with other functions. Also, external diagrams usually more succinct. For example, instead of Figure 6.2, we could have drawn the same thing in the body of the text: $X \xrightarrow{f} Y$.

A function may sometimes also be called a map, or a transformation.
Example 7:

The trigonometric functions $\sin : \mathbb{R} \to \mathbb{R}$ and $\cos : \mathbb{R} \to \mathbb{R}$ are indeed functions because, every $\theta \in \mathbb{R}$, determines $\sin(\theta) \in \mathbb{R}$, and similarly for $\cos$. On the other hand, $\tan$ is not a function $\mathbb{R} \to \mathbb{R}$ because, for example, there is no such thing as $\tan\left(\frac{\pi}{2}\right)$. To deal with such things, $\tan$ is sometimes said to be partial function. We won’t deal with partial functions as a distinct concept in this text.

The reader will not have any trouble thinking of many other natural examples of functions $f : \mathbb{R} \to \mathbb{R}$.

Rules

To define a function, we must specify a set $X$ to be the domain, a set $Y$ to be the codomain, and for each $x \in X$, an element $f(x) \in Y$. Though there are many ways to do this, we will use a notational style that is close to what one finds in some programming languages. For example, to define the function from $\mathbb{N}$ to $\mathbb{N}$ that adds one then squares the input, we may write

$$s(n : \mathbb{N}) = (n + 1)^2 : \mathbb{N}.$$  

This specifies that $s(0) = (0 + 1)^2 = 1$, $s(1) = (1 + 1)^2 = 4$, $s(2) = (2 + 1)^2 = 9$, and so on.

The variable appearing in a rule like this is a placeholder. Its name does not matter. So

$$s(x : \mathbb{N}) = (x + 1)^2 : \mathbb{N}$$

defines the same function $s$ by the same rule.

Very frequently, the domain and codomain are either clear from the context of discussion or can easily be inferred. In those situations, writing a definition as

$$s(x) = (x + 1)^2$$

suffices.

Sometimes, it is helpful to be able to specify a rule without giving the function a name. We can do that by writing something like $x \mapsto x^3 : \mathbb{N} \to \mathbb{N}$, or just $x \mapsto x^3$ if the codomain and codomain are understood. To give a name and define a rule all in one place, we might also write

$$c = (x \mapsto x^3)$$
but that is not any easier to read than
\[ c(x) = x^3. \]

Consider functions \( s: \mathbb{N} \to \mathbb{N} \) and \( t: \mathbb{N} \to \mathbb{N} \) defined by the rules

\[
\begin{align*}
  s(n) &= (n + 1)^2 \\
  t(n) &= n^2 + 2n + 1.
\end{align*}
\]

Are these functions equal? The steps of calculation are quite different. In the first, one simply finds the successor of \( n \) and squares that. In the second, one squares the value \( n \), doubles \( n \), adds those two results and then adds one to the sum.

Though calculation of \( s(n) \) and \( t(n) \) involve very different steps, they arrive at the same results because for all \( n \), it is the case that \((n + 1)^2 = n^2 + 2 \cdot n + 1\). Putting this in function notation, \( s(n) = t(n) \) for every \( n \in \mathbb{N} \). So the attribute of a natural number “square of the successor” is the same as the attribute “sum of the square and the double and one”.

Regarding \( s \) and \( t \) as mathematical attributes, we conclude they are equal. This leads to a principle that gives a general criterion for equality of functions.

**Principle 2: Function Extensionality**

For functions \( f: X \to Y \) and \( g: X \to Y \),

\[ f = g \quad \text{if and only if} \quad f(x) = g(x) \quad \text{for all} \quad x \in X. \]

### 6.3 Relations

Relations correspond roughly to transitive verbs: “4 precedes 5”, “Jim likes waffles”, “The line \( y = 3x + 2 \) intersects the line \( y = 2x - 1 \)”, and so on.

**Vocabulary 5: Basic Vocabulary for Relations**

For sets \( X \) and \( Y \), a relation between \( X \) and \( Y \) is an entity \( R \) so that for each \( x \in X \) and each \( y \in Y \), either the relation “holds” or does not “hold”. We typically write \( x \mathrel{R} y \) and \( x \not\mathrel{R} y \) when the relation holds or
does not hold. A relation between $X$ and $X$ is called a relation on $X$.

For this text, we write $R: X \leftrightarrow Y$ to indicate that $R$ is a relation between $X$ and $Y$, though this notation is not universally known.

As with functions, we say that $X$ is the domain and $Y$ is the codomain of a relation $R: X \leftrightarrow Y$. Sometimes, instead of domain and codomain, these are called the source and target.

For our purposes, when we say that $R$ is a relation, we must have explicitly in mind the domain $X$ and codomain $Y$.

**Example 8:**

For any two natural numbers $m$ and $n$, either $m < n$ or not. So $<$ ("less than") is a relation on $N$. Likewise, for any two real numbers $x$ and $y$, either $x < y$ or not. So $<$ is a relation on $R$. But it is important to bear in mind that, even though we use the same symbol, these are distinct relations. For example, when we are dealing with natural numbers, it makes sense to ask for the smallest $n$ satisfying $5 < n$ — namely, 6. But for real numbers, there is no smallest $y$ satisfying $5 < y$. So the "less than" relation on natural numbers and the "less than" relation on real numbers behave very differently.

**Example 9:**

We can say that $m$ divides $n$ if $m \cdot d = n$ for some natural number $d$. So “divides” is a relation on natural numbers. For example, 5 divides 10, but 5 does not divide 12. We will investigate divisibility in detail in Chapter 12.

**Example 10:**

Suppose we have defined a set $\mathbb{R}[x]$ that consists of all polynomial functions having real coefficients. So $p(x) = x^2 + 3x + 1$ is in $\mathbb{R}[x]$, and so on. Then we can define a relation $is$-$root$: $\mathbb{R} \leftrightarrow \mathbb{R}[x]$ by

$$a \ is$-$root$ $p \quad if \ and \ only \ if \quad 0 = p(a).$$
For the example $p$ above, there is no (real number) root. But for $q(x) = x^2 - 5x + 6$, there are two roots: $2 \text{ is-root } q$ and $3 \text{ is-root } q$, but for example $4 \text{ is-root } q$.

This example illustrates that a relation can hold between data of one type (real numbers) and another type (real polynomials).

Suppose $P$ is a set corresponding to people. Then we can imagine two relations likes and knows on $P$. For example, John likes Pat and Pat knows Kelly. Suppose it is impossible to like someone without knowing the person. We can say this by asserting that to likes implies to knows. This is similar to subset inclusion. This leads to a definition.

**Definition 5: Subrelations**

Suppose $R: X \to Y$ and $S: X \to Y$ are relations. We say that $R$ is a subrelation of $S$ if it is the case that for every $a \in X$ and $b \in Y$, if $aRb$ then $aSb$.

A very simple example of this is that for natural numbers $x$ and $y$, if $x < y$ then $x \leq y$. So $<$ is a subrelation of $\leq$. Just like sets and functions, this leads to a criterion for equality.

**Principle 3: Relational extensionality**

Suppose $R: X \to Y$ and $S: X \to Y$ are relations. Then $R = S$ if and only if $R$ is a subrelation of $S$ and $S$ is a subrelation of $R$.

### 6.4 Chickens and eggs

Though we have discussed the three separate concepts of sets, functions and relations in this chapter, you may have noticed that it looks like we needed sets first, just in order to make sense of domain and codomain of functions and relations. In fact, with a bit more development (discussed in Chapter ??), we can interpret relations as being special kinds of sets and functions as being special kinds of relations. Most mathematicians over more than the last century have been taught to think this way.

If our main concern is to limit the number of kinds of things in mathematics, then a sets-first view seems pretty reasonable.
But it turns out that with a bit more effort, we could also have taken relations to be basic, defining functions and sets in terms of relations. Or we could have taken functions to be basic, defining relation and sets in terms of functions.

Historically (for well over the last century), the sets-first view has been the default. But it seems to me that putting sets, functions and relations on an even footing is a better way to see their distinct roles. In later chapters, we will have chances to concentrate on one or the other.

Exercises

Suppose B refers to the set of books in your university library. Each book has a year of publication. So we might refer to this as \( \text{year} : B \to \mathbb{N} \). Suppose also that S refers to the set of students at the university.

54. Describe three other functions from B to \( \mathbb{N} \) that might be part of the library catalogue.


56. Describe two different relations from students to books.
Functions and sets constitute what is called a category. We will encounter other categories later. For now we use the word informally. For sets and functions, the category structure is specified by two standard rules. We will look at how these rules can be described in diagrams.

**Chapter Goals**

In this chapter, we establish basic ways that functions behave and how they can be combined. Wiring diagrams provide a useful way to reason about functions.

---

**Principle 4: Composition and Identity Rules**

For any two functions \( f : W \rightarrow X \) and \( g : X \rightarrow Y \), there is a function from \( W \) to \( Y \) called \( g \) following \( f \) or \( g \) after \( f \). It is usually denoted by

\[
g \circ f : W \rightarrow Y,
\]

and is defined by the rule

\[
x \mapsto g(f(x)).
\]

For any set \( X \), there is a function on \( X \) called the identity on \( X \). It is usually denoted by \( \text{id}_X : X \rightarrow X \). This is defined by the rule

\[
x \mapsto x.
\]

Notice that the codomain of \( f \) must agree with the domain of \( g \).

Sometimes \( \circ \) is omitted. So \( gf \) means the same as \( g \circ f \).
LEMMA 14: Sets and functions form a category

Sets and functions form a category via composition and identity functions. Specifically,

- For functions $f: W \to X$, $g: X \to Y$ and $h: Y \to Z$,
  $$h \circ (g \circ f) = (h \circ g) \circ f.$$  

- For function $g: X \to Y$,
  $$g = g \circ \text{id}_X \quad \text{and} \quad g = \text{id}_Y \circ g.$$

PROOF: The proof of this is a very simple exercise using Function Extensionality. □

Exercises

57. Prove that Lemma 14 is a consequence of Principles 2 and 4.

7.1 Wiring diagrams

It can be helpful to picture a function as a machine that transforms inputs to outputs. We can depict function $f: X \to Y$ and $g: Y \to Z$ like so:

![Figure 7.1: Two wiring diagrams depicting functions](image)

Diagrams like these are called *wiring diagrams* or *string diagrams*. Informally, from an input $a \in X$ on the input wire of the left diagram, the box $f$ produces an element $f(a) \in Y$ on the output wire.
The identity function $\text{id}_X : X \to X$ is the function that does nothing to an input. So we will generally draw it as a simple wire: $\underline{X}$. That is, we don’t bother to draw a little box labelled $\text{id}_X$.

Composition of functions $f : X \to Y$ and $g : Y \to Z$ is just a matter of connecting wires as in Figure 7.2.

![Figure 7.2: A string diagram for $g \circ f$.](image)

We are permitted to connect an output wire to an input wire as long as they agree about the label. This enforces the idea that a wire labelled $X$ carries data of the type $X$.

You might think of it this way. Some wires are 120 volt electrical supply cables, some wires are USB cables. You can connect a supply cable output to a supply cable input (think of a wall socket). Or you can connect a USB output to a USB input (think of connecting a thumb drive to your laptop). But of course, you cannot connect a USB to a supply wall socket. (Well, you can. But it is not a good idea).

The associativity of composition is reflected in diagrams by the fact that we can simply connect additional diagrams. Figure 7.3 could be constructed in two ways: first connect $f$ to $g$, then connect that to $h$ (resulting in $h \circ (g \circ f)$); or first connect $g$ to $h$, then connect $f$ to that (resulting in $(h \circ g) \circ f$). Associativity says the result does not depend on that order.

![Figure 7.3: Composition is associative.](image)

We do not need to distinguish between $h \circ (g \circ f)$ or $(h \circ g) \circ f$.

Feeding an input $a \in X$ into Figure 7.3, results if $f(a)$ on the second wire, and $g(f(a))$ on the third wire and finally $h(g(f(a)))$ on the output wire. So it is a good depiction of the transformation $h \circ g \circ f$.

How should we depict addition? Figure 7.4 is an obvious thing to try.
With diagrams like these, we can draw complicated compositions of functions, and can even draw pictures of equations. For example, (omitting the label \( \mathbb{N} \) on all wires) Figure 7.5 asserts that addition is associative.

If we name the three input wires \( x, y \) and \( z \), then the left diagram describes the output \((x + y) + z\). The right diagram describes the output \(x + (y + z)\). The equation asserts that \((x + y) + z = x + (y + z)\).

In the next section, we address a problem with wiring diagrams like this. Namely, the vocabulary for functions supposes that a function has a domain and a codomain, both of which are sets. Figure 7.4 does not fit how we talk about functions because + has two inputs, not one.

### 7.2 Products

The solution to dealing with multiple argument functions like addition is to think carefully about the wiring of wiring diagrams, and to allow sets that encode data on bundled wires.

Suppose \( X \) and \( Y \) are sets. Informally, let us write \( X \times Y \) to denote a bundled wire consisting of \( X \) and \( Y \). So if we were to “zoom in” on \( X \times Y \), we would see something like this:

The idea is, roughly, that a cable \( X \times Y \) consists of two separate wires \( X \) and \( Y \).
So the basic addition diagram (Figure 7.4) could be drawn as in Figures 7.7 or 7.6.

To bring this into our system, we need to understand $X \times Y$ as an actual set.
**Principle 5: Cartesian Products**

For any sets $X$ and $Y$, there is a set, denoted by $X \times Y$, that consists of all pairs $(x, y)$ where $x \in X$ and $y \in Y$. Moreover, there are functions

$$X \xrightarrow{\text{pr}_{X,Y}} X \times Y \xrightarrow{\text{pr}_{X,Y}'} Y,$$

defined by

$$\text{pr}_{X,Y}(x, y) = x \quad \text{and} \quad \text{pr}_{X,Y}'(x, y) = y.$$

For any two functions $X \xleftarrow{f} C \xrightarrow{g} Y$, there is a function, $(f, g): C \to X \times Y$ defined by

$$(f, g)(c) = (f(c), g(c)).$$

We can draw diagrams for $\text{pr}$ and $\text{pr}'$ by capping one or the other of two parallel wires.

Figure 7.8: Projection diagrams take input from $X \times Y$ and discard one of the inputs.

Consider the two functions in Figure 7.9. According to Principle 5, there is also a single function $(f, g): C \to X \times Y$ that captures the idea of using $f$ and $g$ on the same inputs. This can be depicted as in Figure 7.10.

Composing Figure 7.10 with one of the projection diagrams from Figure 7.8, yields Figure 7.11.

Figure 7.11 suggests that the result of projection following pairing should satisfy the equations

$$\text{pr}_{X,Y} \circ (f, g) = f \quad \text{and} \quad \text{pr}_{X,Y}' \circ (f, g) = g.$$
Figure 7.9: Two functions with the same domain.

Figure 7.10: A single function \( \langle f, g \rangle \), pairing \( f \) and \( g \). An input \( c \) is copied as input to \( f \) and to \( g \).

Figure 7.11: Composing \( \langle f, g \rangle \) after \( \text{pr}_{X,Y} \). An input \( c \) produces \( f(c) \) and \( g(c) \) on the \( X \) and \( Y \) wires, and ignores \( g(c) \). The result is \( f(c) \).
have

\[ \text{pr}_{X,Y} \circ \langle f, g \rangle (c) = \text{proj}_{X,Y}(\langle f, g \rangle (c)) \]
\[ = \text{pr}_{X,Y}(f(c), g(c)) \]
\[ = f(c) \]

So by function extensionality (Principle 2), \( \text{pr}_{X,Y} \circ \langle f, g \rangle = f \). An identical calculation shows that \( \text{pr}^{' \prime}_{X,Y} \circ \langle f, g \rangle = g \).

Also, consider composing projections with pairing in the other order. That is, a function \( h : C \to X \times Y \) can be drawn as in Figure ref.

![Figure 7.12: A function with codomain \( X \times Y \).](image)

Composing \( h \) with the two projections produces the diagrams in Figure 7.13

![Figure 7.13: The diagrams \( \text{pr}_{X,Y} \circ h \) and \( \text{pr}^{' \prime}_{X,Y} \circ h \). We have omitted the dashed boxes indicating the two projections.](image)

Finally, pairing the diagrams in Figure 7.13 yields Figure 7.14.

![Figure 7.14: The diagram \( \langle \text{pr}_{X,Y} \circ h, \text{pr}^{' \prime}_{X,Y} \circ h \rangle \). Again, we have omitted dashed boxes.](image)
This diagram equals $h$ because for any $c \in C$,

$$\langle \text{pr}_{X,Y} \circ h, \text{pr}'_{X,Y} \circ h \rangle(c) = (\text{pr}_{X,Y}(h(c)), \text{pr}'_{X,Y}(h(c))) = h(c)$$

Now consider two unrelated functions $d: W \to Y$ and $e: X \to Z$ as in Figure 7.15. These can be assembled into a single function $(d \times e): W \times X \to Y \times Z$.

In a formula, $d \times e = (d \circ \text{pr}_{W,X}, e \circ \text{pr}'_{W,X})$, so

$$(d \times e)(x, y) = (d \circ \text{pr}_{W,X}(x, y), e \circ \text{pr}'_{W,X}(x, y)) = (d(x), e(y))$$

### 7.3 Thunks and Empty Diagrams

In computing, a thunk is a function that does not take any input, but that is nevertheless a function. In a wiring diagram, we expect a thunk to look like this:

because it has no input. To bring this idea into the mathematics of sets and functions, we need to consider the domain of $t$. But $t$ is not meant to depend on input. Putting it more positively, $t$ is meant to depend on no input.

Remember that $X \times Y$ allowed us to deal with two inputs. Evidently, we need a way to think about an empty product.
PRINCIPLE 6: A terminal set exists.

There is a set \( 1 \) that consists of exactly one element. The particular element is not critical. So for our purposes, we will let \( 1 = \{ \bullet \} \).

The set \( 1 \) is a terminal set. That is, for any set \( C \), there is exactly one function \( \Diamond_C : C \to 1 \). It is defined by \( \Diamond_C(c) = \bullet \).

It helps to think of \( 1 \) as empty product. We we noted, it is helpful to depict \( X \times Y \), sometimes as two parallel wires and sometimes as a single wire labelled \( X \times Y \). Likewise, sometimes it is helpful to depict \( 1 \) as an empty diagram (no wires), and sometimes as a single wire labelled \( 1 \).

Usually, we will depict a thunk (a function with domain \( 1 \)) as in the diagram at the beginning of this section. So \( 1 \) is drawn as no wires. A thunk \( t : 1 \to X \) can only do one thing: \( t(\bullet) \) is an element of \( X \). Also, every element \( a \in X \) should correspond to a thunk — namely, the function from \( 1 \) to \( X \) that simply produces \( a \). This leads to another principle.

PRINCIPLE 7: Elements are thunkable.

For any set \( X \), and any element \( a \in X \), there is a function \( \hat{a}_X : 1 \to X \) defined by \( \hat{a}_X(\bullet) = a \).

In a wiring diagram, we draw a thunk, perhaps using a special symbol, as

\[
\hat{a} \quad \text{Y}
\]

where \( a \) is an element of \( Y \).

7.4 Parametric Functions

Suppose we wish to define a function \( \text{clock} : T \to \text{string} \) that turns a computer’s time (here the set \( T \) represents the data type of time stamps on a Unix clock, which are not easily read by us humans) into
a string of human-readable symbols. So for example $\text{clock}(2340234291)$ may produce the string 12:00:42.

To make this function really useful, it ought to know what time zone the user wants to use. So really, we could regard $\text{clock}$ as a function that takes two arguments. The first argument is a time zone code, the second is a UNIX time. So we could draw our diagram of the clock as in Figure 7.16.

Figure 7.16: A clock function with time zone parameter

![Diagram of clock function with time zone parameter](image)

Figure 7.16 is not really different than any other diagram with two inputs. We have just moved the wire representing time zone ($Z$) to emphasize that we want to think of $\text{clock}$ mainly as a function $T \to \text{String}$ with a parameter that sets the time zone.

There are many situations in which we wish to think about parametric functions like this. In calculus, for example, indefinite integration takes a real integrable function $f$ and returns a family of functions $F(x) + C$ where $f(x) = \frac{dF(x)}{dx}$. So $\int f(x)dx$ is not a function, but is a family of functions parameterized by the constant $C$. Integration only defines an actual function once we fix $C$. This leads us to introduce a convenient definition.

**Definition 6:**

For sets $P$, $X$ and $Y$, a $P$-parametric function from $X$ to $Y$ is a function $f: P \times X \to Y$.

In other words, $f$ depends on a parameter $p \in P$ and an input $x \in X$. There is no mathematical difference between a (simple) function from $P \times X$ to $Y$ and a $P$-parametric function from $X$ to $Y$. It is a matter of what we wish to emphasize.
Suppose we are given a $Q$-parametric function $f: Q \times X \to Y$ and a simple function $c: P \to Q$, then we can change parameters by assembling these into a $P$-parametric function as in Figure 7.17.

This leads to the following definition.

**Definition 7: Universal Parameter Set**

For sets $X$ and $Y$, a universal parameter set is a set $E$ equipped with an $E$-parametric function from $X$ to $Y$ — that is, $e: E \times X \to Y$ — so that every $P$-parametric function $h: P \times X \to Y$ is obtained uniquely from $e$ by a change of parameters. In more detail, there is a unique function $h^*: P \to E$ so that $h = e \circ (h^* \times \text{id}_X)$. Figures 7.18 and 7.19 illustrate how $h^*$, $e$, and $h$ are related.

In general universal parameter objects are called exponents. This will make sense later.
The definition of exponential sets does not guarantee they exist. For that, we either need a principle.

**Principle 8:** Universal parameter sets exist.

For any two sets $X$ and $Y$, there is a universal parameter set (also called exponent) denoted by $Y^X$ with $Y^X$-parametric function denoted by

$$
ev_{X,Y} : Y^X \times X \to Y.$$ 

For a $P$-parametric function $h : P \times X \to Y$, the construction from $h$ to $h^\dagger : P \to Y^X$ is called **currying**, after the 20th century logician Haskell Curry. In his honor, sometimes it is written $\text{curry}(h)$.

Now consider a function $f : X \to Y$. We can, somewhat artificially, think of it as a 1-parametric function, $f \circ \text{pr}_{1,X}$, as in Figure 7.4.

The “curried” version of this is $(f \circ \text{pr}_{1,X})^\dagger$. This is a function from 1 to $Y^X$. As we noted earlier, such functions are “thunks”, corresponding exactly to one element of $Y^X$. We may denote this element $\lceil f \rceil$. 

Figure 7.19: An exponential set $E$ for $X$ and $Y$ acts as a universal parameter set. Any other parametric function is obtained by change of parameter.
and call it the name of \( f \). Thus, for any function \( f : X \to Y \), the element \( \langle f \rangle \in Y^X \) is the unique element for which \( \hat{\langle f \rangle} = (f \circ \text{id}_X)\). 

Conversely, suppose \( \phi \in Y^X \). Then \( \hat{\phi} : 1 \to Y^X \) can be used as a change of parameters. So \( ev \circ (\hat{\phi} \times \text{id}_X) \) is a \( 1 \)-parametric function from \( X \) to \( Y \). And so \( ev_{X,Y} \circ (\hat{\phi} \circ \square_X, \text{id}_X) \) is a function from \( X \) to \( Y \). Call this the function named by \( \phi \), and denote it by \( \phi \). So each element of \( Y^X \) determines a function \( X \to Y \), and each such function determines an element of \( Y^X \). We can gather these ideas into another definition.

**Definition 8: Names of functions**

For any function \( f : X \to Y \), let \( \langle f \rangle \in Y^X \) denote the unique element of \( Y^X \) so that \( \hat{\langle f \rangle} = (f \circ \text{id}_X)\). We call \( \langle f \rangle \) the name of \( f \).

For any \( \phi \in Y^X \), let \( \hat{\phi} : X \to Y \) denote the unique function that is named by \( \phi \).

Now it is an exercise to show that

\[
f = \hat{\phi}
\]

if and only if

\[
\langle f \rangle = \phi.
\]

In concrete terms of elements: \( ev(\langle f \rangle, x) = f(x) \) for any function \( f : X \to Y \). And for any \( P \)-parametric function \( h : P \times X \to Y \), any \( p \in P \), the element \( h^\dagger(p) \in Y^X \) is the name of a function so that that \( h^\dagger(p)(x) = h(p, x) \). It is sometimes convenient to have an explicit name for the function \( X \to Y \) that is named by \( h^\dagger(p) \). The common notation is \( \lambda x \cdot h(p, x) \). That is, \( \lambda x \cdot h(p, x) = h^\dagger(p) \).

This means that if we intend to define a function \( F : P \to Y^X \), we can specify \( F(p)(x) \in Y \) for each \( p \in P \) and \( x \in X \). Or we can spell out just require \( F(p) = \langle \lambda x \cdot f(p, x) \rangle \) for some choice of \( f : P \times X \to Y \).

**Example 11:**

A law of exponents for natural numbers says that \( (m^n)^p = m^{np} \). Since we are re-using the word “exponent” and the notation \( Y^X \), it seems reasonable to ask if a similar law holds. That is, it is the case that \( (Y^X)^W \) and \( Y^{X \times W} \) are the same sets?

It turns out that the answer is, strictly, no. But the two sets are completely interchangible. Specifically, there are two functions...
F: \( (Y^X)^W \to Y^{X \times W} \) and G: \( Y^{X \times W} \to (Y^X)^W \) that are mutual inverses of each other.

To define F, we must specify \( F(\phi) \in Y^{X \times W} \) for each \( \phi \in (Y^X)^W \). So it suffices to specify \( F(\phi)(x, w) \in Y \) for each \( x \in X \) and \( w \in W \). But \( \phi \) is itself the name of some function. So \( \phi(w) \) is an element of \( Y^X \), and \( \phi(w)(x) \in Y \). So we may indirectly define F by requiring

\[
F(\phi)(x, w) = \phi(w)(x).
\]

Similarly, we can define G by specifying for each \( \psi \in Y^{X \times W} \) the behavior of \( G(\psi) \in (Y^X)^W \). Again, this amounts to defining \( G(\psi)(w) \in Y^X \) for each \( w \in W \), and this amounts to defining \( G(\psi)(w)(x) \in Y \) for each \( x \in X \). But recalling that \( \psi \) is a name for a function from \( X \times W \) to \( Y \), we see immediately that

\[
G(\psi)(w)(x) = \psi(x, w)
\]

is the only sensible definition to try.

Now we check whether F and G are inverses of each other.

\[
G(F(\phi))(w)(x) = F(\phi)(x, w) = \phi(w)(x)
\]

Since this is true for all \( x \),

\[
G(F(\phi))(w) = \phi(w)
\]

So \( G(F(\phi)) = \phi \). Similarly,

\[
F(G(\psi))(x, w) = G(\psi)(w)(x) = \psi(x, w).
\]

So \( F(G(\psi)) = \psi \).

---

**Exercises**

58. Show that for any sets \( W, X \) and \( Y \), there are functions \( F: (W \times X)^Y \to W^Y \times W^X \) and \( G: W^Y \times W^X \to (W \times X)^Y \) that are inverses of each other.

59. Show that for any set \( X \), there is exactly one element of \( 1^X \). [This also shows that \( 1^X \) is a terminal set.]

60. Show that for any set \( X \), there are functions \( F: X \to X^1 \) and \( G: X^1 \to X \) that are inverses of each other.
7.5 External Diagrams

Wiring diagrams are useful for getting the idea of function composition, but they are awkward for reasoning about equalities of functions because we need to draw two (possibly complicated) diagrams just to state that they represent equal functions. That’s fine for small examples, but is not helpful for more complicated situations involving many different functions.

When we are mainly concerned with how functions interact, an individual function can be depicted simply as $X \xrightarrow{f} Y$. A composition of functions can be depicted as in

$$\begin{align*}
X & \xrightarrow{f} Y \\
g \circ f & \downarrow \\
Z &
\end{align*}$$

We do not really need to draw $g \circ f$ as a separate arrow because the path from $X$ to $Y$ to $Z$ is already an implicit part of the diagram. So the simpler diagram

$$\begin{align*}
X & \xrightarrow{f} Y \\
g & \downarrow \\
Z &
\end{align*}$$

conveys the same information, namely, that $f: X \to Y$ and $g: Y \to Z$ are functions and therefore, $g \circ f: X \to Z$ is too. Notice that we also did not bother to draw the identity functions for $X$, $Y$ and $Z$. If we did draw them, they would appear as “loops”.

The diagram

$$\begin{align*}
W & \xrightarrow{f} X \\
h & \downarrow \\
Y & \xrightarrow{g} Z \\
k &
\end{align*}$$

depicts ten functions all together: four identity functions, the named functions $f$, $g$, $h$, and $k$, and the two composite functions $g \circ f$ and $k \circ h$. We say that such a diagram commutes if $g \circ f = k \circ h$. More generally, to assert that a diagram commutes means all the compositions depicted implicitly in the diagram that can be equal are equal. So a commutative diagram is a graphical depiction of equations. This takes some getting used to, but it is worth the practice.

Consider the Law of Associativity for composition. For composable
functions, we can draw the basic situation as

\[ W \xrightarrow{f} X \xrightarrow{g} Y \xrightarrow{h} Z \]

Including the compositions \( g \circ f \) and \( h \circ g \), we get

\[ W \xrightarrow{f} X \xrightarrow{h \circ g} Z \]

\[ g \circ f \xrightarrow{g \circ f} Y \xrightarrow{h \circ g} Z \]

Associativity says that this diagram commutes. Likewise, the Identity Laws say that the diagrams

\[ \begin{array}{c}
\begin{array}{c}
X \\
| \\
f \\
| \\
Y
\end{array} \\
\begin{array}{c}
f \downarrow \\
| \\
Y
\end{array} \\
\begin{array}{c}
Y \\
| \\
\text{id}_Y \\
| \\
Y
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
X \\
| \\
id_X \\
| \\
X
\end{array} \\
\begin{array}{c}
id_X \\
| \\
X
\end{array} \\
\begin{array}{c}
f \\
| \\
f
\end{array}
\end{array} \]

both commute. We will encounter many other examples later.

Consider the functions \( f: \mathbb{N} \to \mathbb{N}, g: \mathbb{N} \to \mathbb{N}, h: \mathbb{N} \to \mathbb{N} \) and \( k: \mathbb{N} \to \mathbb{N} \) defined by

\[
\begin{align*}
f(n) &= n + 1 \\
g(n) &= n^2 \\
h(n) &= 2n \\
k(n) &= n^3
\end{align*}
\]

We can form a diagram depicting \( g \circ f \) and \( k \circ h \) as in

\[ \begin{array}{c}
\begin{array}{c}
\mathbb{N} \\
| \\
f \\
| \\
\mathbb{N}
\end{array} \\
\begin{array}{c}
h \\
| \\
\mathbb{N} \\
| \\
\mathbb{N}
\end{array} \\
\begin{array}{c}
g \\
| \\
\mathbb{N}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\mathbb{N} \\
| \\
k \\
| \\
\mathbb{N}
\end{array} \\
\begin{array}{c}
h \\
| \\
\mathbb{N} \\
| \\
\mathbb{N}
\end{array} \\
\begin{array}{c}
g \\
| \\
\mathbb{N}
\end{array}
\end{array} \]

but this diagram does not commute because, for example, \( g \circ f(2) = 9 \neq 64 = k \circ h(2) \). The mere fact that we can draw such a diagram and that the paths corresponding to \( g \circ f \) and \( k \circ h \) start and end at the same places is not a guarantee the diagram commutes.
Exercises

61. For each of the following pairs of functions \( \mathbb{N} \to \mathbb{N} \), determine whether they are equal and explain why or why not.

(a) \( f(n) = 2n + 3 \) and \( g(m) = 2m + 3 \)
(b) \( f(n) = 2^{n+1} - 1 \) and \( g(n) = \sum_{i=0}^{n} 2^i \)
(c) \( f(n) = n^2 + 5n + 6 \) and \( g(n) = (n + 3)(n + 2) \)
(d) \( f(n) = n^4 - 10n^3 + 35n^2 + 50n + 24 \) and \( g(n) = 24 \)

62. Let \( \mathbb{R} \) denote the set of all real numbers. Explain why \( \tan \) (the tangent “function”) does not actually define a function from \( \mathbb{R} \) to \( \mathbb{R} \).

63. Suppose the following functions exist: \( f: W \to X \), \( g: X \to Y \), \( a: W \to Z \), \( b: Y \to Z \). Draw a commutative diagram asserting that \( b \circ g \circ f = a \).

64. Suppose the following functions exist: \( f: C \to A \), \( g: C \to B \), \( h: C \to P \), \( p: P \to A \) and \( q: P \to B \). Draw a single commutative diagram asserting that \( f = p \circ h \) and \( g = q \circ h \).

65. Define functions from \( \mathbb{N} \) to \( \mathbb{N} \) as follows:

\[
\begin{align*}
    f(n) &= n + 1 \\
    g(n) &= n^2 \\
    h(n) &= n^2 + 2n
\end{align*}
\]

Draw a diagram depicting the compositions \( g \circ f \) and \( f \circ h \). Does the diagram commute?

7.6 Tabulating a Function

Recall the sets \textsc{deck}, \textsc{rank} and \textsc{suit} from Chapter 6. Given a card \( c \in \textsc{deck} \), there is evidently an assignment of a rank. For example, the rank of \( 5\heartsuit \) is 5. Likewise, there is an assignment of suit of any card.

The rules defining functions \textsc{rank} and \textsc{suit} are obvious, but to make them completely explicit, we can build a table:

We do not really need the table in Figure 7.1 because all the values are easily predictable. The cards themselves are systematically built from all possible pairings of a rank and a suit. Nevertheless, this
sort of tabulation of functions can be useful in more complicated situations.

Suppose we have dealt the cards to four players. In many games, the players are named N, W, S, and E. So let \( \text{PLAYER} = \{N, W, S, E\} \). Now the result of dealing the cards is an assignment of a player to each card (spelling out which card is held by which player).

For example, Table 7.2 shows one way the cards might be dealt.
<table>
<thead>
<tr>
<th>$c \in \text{DECK}$</th>
<th>$\text{holder}(c)$</th>
<th>$c \in \text{DECK}$</th>
<th>$\text{holder}(c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A♠</td>
<td>N</td>
<td>A♦</td>
<td>S</td>
</tr>
<tr>
<td>2♠</td>
<td>S</td>
<td>2♦</td>
<td>E</td>
</tr>
<tr>
<td>3♠</td>
<td>E</td>
<td>3♦</td>
<td>W</td>
</tr>
<tr>
<td>4♠</td>
<td>N</td>
<td>4♦</td>
<td>W</td>
</tr>
<tr>
<td>5♠</td>
<td>W</td>
<td>5♦</td>
<td>N</td>
</tr>
<tr>
<td>6♠</td>
<td>S</td>
<td>6♦</td>
<td>S</td>
</tr>
<tr>
<td>7♠</td>
<td>E</td>
<td>7♦</td>
<td>E</td>
</tr>
<tr>
<td>8♠</td>
<td>W</td>
<td>8♦</td>
<td>S</td>
</tr>
<tr>
<td>9♠</td>
<td>N</td>
<td>9♦</td>
<td>N</td>
</tr>
<tr>
<td>10♠</td>
<td>S</td>
<td>10♦</td>
<td>E</td>
</tr>
<tr>
<td>J♠</td>
<td>E</td>
<td>J♦</td>
<td>E</td>
</tr>
<tr>
<td>Q♠</td>
<td>E</td>
<td>Q♦</td>
<td>E</td>
</tr>
<tr>
<td>K♠</td>
<td>W</td>
<td>K♦</td>
<td>S</td>
</tr>
<tr>
<td>A♣</td>
<td>N</td>
<td>A♦</td>
<td>N</td>
</tr>
<tr>
<td>2♠</td>
<td>S</td>
<td>2♥</td>
<td>E</td>
</tr>
<tr>
<td>3♠</td>
<td>W</td>
<td>3♥</td>
<td>W</td>
</tr>
<tr>
<td>4♠</td>
<td>N</td>
<td>4♥</td>
<td>N</td>
</tr>
<tr>
<td>5♠</td>
<td>W</td>
<td>5♥</td>
<td>E</td>
</tr>
<tr>
<td>6♠</td>
<td>W</td>
<td>6♥</td>
<td>W</td>
</tr>
<tr>
<td>7♠</td>
<td>W</td>
<td>7♥</td>
<td>N</td>
</tr>
<tr>
<td>8♠</td>
<td>N</td>
<td>8♥</td>
<td>S</td>
</tr>
<tr>
<td>9♠</td>
<td>S</td>
<td>9♥</td>
<td>S</td>
</tr>
<tr>
<td>10♠</td>
<td>E</td>
<td>10♥</td>
<td>E</td>
</tr>
<tr>
<td>J♠</td>
<td>W</td>
<td>J♥</td>
<td>W</td>
</tr>
<tr>
<td>Q♣</td>
<td>N</td>
<td>Q♥</td>
<td>S</td>
</tr>
<tr>
<td>K♣</td>
<td>S</td>
<td>K♥</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 7.2: A deal of four bridge hands
Exercises

66. Confirm or deny that the function specified in Table 7.2 is legal.

For a less uniform example, suppose we have a set corresponding to six students \(S = \{s_1, s_2, s_3, s_4, s_5, s_6\}\). Each student has a first name and a last name. So we can specify their names as in Table 7.3:

<table>
<thead>
<tr>
<th>(x \in S)</th>
<th>forename((x))</th>
<th>family((x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1)</td>
<td>Carol</td>
<td>Binkley</td>
</tr>
<tr>
<td>(s_2)</td>
<td>Pat</td>
<td>Patterson</td>
</tr>
<tr>
<td>(s_3)</td>
<td>Kelly</td>
<td>Treebaum</td>
</tr>
<tr>
<td>(s_4)</td>
<td>Jamie</td>
<td>Doodle</td>
</tr>
<tr>
<td>(s_5)</td>
<td>Kelly</td>
<td>Green</td>
</tr>
<tr>
<td>(s_6)</td>
<td>Violet</td>
<td>Green</td>
</tr>
</tbody>
</table>

Table 7.3: Names of students in \(S\)

Exercises

67. Provide a table defining a function (call it \(f_0\)) from the set \(A = \{0, 1, 2, 3\}\) to \(B = \{N, S, E, W\}\) and a table defining a function (call it \(g_0\)) from \(B\) to \(A\) so that \(id_A = g_0 \circ f_0\) and \(id_B = f_0 \circ g_0\).

68. Provide a table defining a function (call it \(f_1\)) from the set \(A = \{0, 1, 2, 3\}\) to \(B = \{N, S, E, W\}\) and table defining a function (call it \(g_1\)) from \(B\) to \(A\) so that \(id_A = g_1 \circ f_1\) but \(id_B \neq f_1 \circ g_1\).

69. Provide a table defining a function (call it \(f_2\)) from the set \(A = \{0, 1, 2, 3\}\) to \(B = \{N, S, E, W\}\) and table defining a function (call it \(g_2\)) from \(B\) to \(A\) so that \(id_A \neq g_2 \circ f_2\) and \(id_B \neq f_2 \circ g_2\).

70. The table in Figure 7.4 does not represent a function \(f\) from the set \(T = \{a, b, c, d\}\) to the set \(W = \{0, 1, 2, 3, 4, 5, 6\}\). Explain why.

<table>
<thead>
<tr>
<th>(x \in T)</th>
<th>(f(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>0</td>
</tr>
<tr>
<td>(b)</td>
<td>1</td>
</tr>
<tr>
<td>(c)</td>
<td>2</td>
</tr>
<tr>
<td>(d)</td>
<td>1</td>
</tr>
<tr>
<td>(a)</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7.4: A table that does not define a function
Relations and Powersets

Binary relations model concepts like “less than”, “belongs to”, “similar to”, “to the left of”, “intersects” and so on. In English, transitive verbs can be approximately modelled as relations. For example in sentences such as “Jerome likes salsa” or “Parthiv likes watermelon” the verb “likes” denotes a relation between persons and things. Relations are just as fundamental to how we think about mathematics as functions.

Recall that a relation $R$ from $X$ to $Y$ answers a yes/no question for each $a \in X$ and each $b \in Y$. That is, either $aRb$ or $a \not R b$.

**Definition 9:** Relational products, diagonals and relation comparison

For any two relations $R: X \leftrightarrow Y$ and $S: Y \leftrightarrow Z$, there is a relational product $(R; S): X \leftrightarrow Z$ sometimes pronounced “$R$ then $S$” or “$R$ followed by $S$” defined by $a R; S c$ if and only if $a R b$ and $b S c$ for some $b \in Y$.

For any set $X$, there is a relation $\Delta_X$ on $X$ called the diagonal satisfying $a \Delta_X b$ if and only if $a = b$.

For relations $R: X \leftrightarrow Y$ and $S: X \leftrightarrow Y$, we say that $R$ entails $S$ or $R$ is a subrelation of $S$ if and only if for every $x \in X$ and $y \in Y$, $x R y$ implies that $x S y$. We will write $R \sqsubseteq S$ when $R$ entails $S$.

For example, suppose we have sets STUDENT, COURSE and PROF representing students, courses and professors, respectively. Then we might have a relation taking: STUDENT $\leftrightarrow$ COURSE representing which students are currently taking which courses. Likewise, we might have a relation taught-by: COURSE $\leftrightarrow$ PROF representing
which courses are currently being taught by which professors. Then
\textit{taking; taught-by} is a relation from students to professors representing
which student is currently taking a course from which professor.

The relation $\text{taking} \circ$ denotes the relation of a course to the students
in that course.

Exercises

71. What does $\text{taking; taking} \circ$ represent?

8.1 \textbf{Order and Extensionality of Relations}

Two relations from $X$ to $Y$ can be compared by entailment. Notice that
$R \Rightarrow R$ is true for any relation $R$. Also if $R \Rightarrow S$ and $S \Rightarrow T$ then $R \Rightarrow T$
Just as with $\subseteq$ on subsets and $\leq$ on natural numbers, we summarize
this by saying that $\Rightarrow$ is reflexive and transitive. Also like subset
inclusion and $\subseteq$, if $R \Rightarrow R'$ and $R' \Rightarrow R$, then the two relations are
really the same, regardless of how they were originally defined.

\textbf{Postulate 4: Relation Extensionality}

For relations $R: X \leftrightarrow Y$ and $R': X \leftrightarrow Y$, if $R \Rightarrow R'$ and $R' \Rightarrow R$, then
$R = R'$.

8.2 \textbf{Sets and relations as a category}

Recall that sets together with functions constitute a \textit{category}. This
means, essentially, that function composition makes sense when the
codomains and domains match correctly, that each set has an identity
function, and that composition is associative when it makes sense.

We can say the same about relations, plus a bit more.
**Lemma 15:** Sets and relations form an ordered category

For any relation \( S : X \leftrightarrow Y \),
\[
S \cdot \Delta_Y = S = \Delta_X \cdot S.
\]

For any two additional relations \( R : W \leftrightarrow X \), and \( T : Y \leftrightarrow Z \),
\[
(R; S) \cdot T = R \cdot (S; T)
\]
Furthermore, for any two relations \( R' : W \leftrightarrow X \) and \( T' : Y \leftrightarrow Z \), if \( R \Rightarrow R' \), then
\[
R \cdot S \Rightarrow R' \cdot S,
\]
and if \( T \Rightarrow T' \), then
\[
S \cdot T \Rightarrow S \cdot T'.
\]

**Proof:** Everything is easily verified by the definition of relation composition. We leave this as an exercise. \( \square \)

The category of sets and relations is typically referred to as \( \text{Rel} \), whereas the category of sets and functions is referred to set \( \text{Set} \).

### 8.3 Depicting relations

A relation from finite set \( X \) to finite set \( Y \) can be represented as a table. Rows of the table represent elements of \( X \); columns, elements of \( Y \). Then putting a mark in row \( a \), column \( b \) indicates \( a R b \). For example, here is a representation of a relation \( \text{likes} \) between a set \( \text{FRIENDS} = \{\} \) and \( \text{DISHES} = \{\} \) indicating which friend likes which dish.

Obviously, the mark (✓) could be any symbol. Also, the blank entries could be some other symbol, so long as we can distinguish the two cases. It can be helpful to think of a relation as a table consisting of 1’s and 0’s instead of ✓’s and blanks.

Because relations form a category, we can also use wiring diagrams and external diagrams exactly as we did for functions. Naturally, one needs to be careful not to confuse functions and relations, because different facts hold for them. Moreover, since relations may be ordered by entailment, an external diagram can express more than just equality. Figure 8.1 illustrates this.
8.4 Relations from Functions

Every function from $X$ to $Y$ determines a relation from $X$ to $Y$, called the graph of the function and is defined as follows.

**Definition 10: Graph of a function**

For a function $f : X \to Y$, define the graph of $f$ to be the relation $\Gamma_f : X \leftrightarrow Y$ given by $a \Gamma_f b$ if and only if $f(a) = b$.

The graphs of functions cooperate with identities and with composition in the following way.

**Lemma 16: Composition of graphs equals graph of composition**

For functions $f : X \to Y$ and $g : Y \to Z$, it is the case that $\Gamma_{g \circ f} = \Gamma_f \circ \Gamma_g$. Moreover, $\Gamma_{id_X} = \Delta_X$ holds for any set $X$.

**Proof:** Suppose $a \Gamma_{g \circ f} c$. By definition there is some $b \in Y$ so that $f(a) = b$ and $g(b) = c$. So $g(f(a)) = c$, and hence $a \Gamma_{g \circ f} c$.

Conversely, suppose $a \Gamma_{g \circ f} c$. Then $g(f(a)) = c$. So $a \Gamma_f f(a)$ and $f(a) \Gamma_g c$. That is, $a \Gamma_f \Gamma_g c$.

Clearly, $\Gamma_{id_X} = \Delta_X$. □

Suppose we have a function $f : X \to Y$. We can consider the composition of $\Gamma_f$ with its opposite. That is, $\Gamma_f ; \Gamma_f^\circ$. For any $a \in X$, $a \Gamma_f f(a)$ and $f(a) \Gamma_f a$ both hold. So $a \Gamma_f ; \Gamma_f^\circ a$. In other words, we have shown that $\Delta_X \subseteq \Gamma_f ; \Gamma_f^\circ$. The only property we needed here is that for each $a \in X$ there is some $b \in Y$ so that $a \Gamma_f b$.

Looking at the opposite composition, suppose $b \Gamma_f^\circ ; \Gamma_f b'$. Then there is some $a \in X$ so that $b = f(a)$ and $f(a) = b'$. So $b = b'$. That
is, $\Gamma_f; f \subseteq \Delta_Y$. The only property of we needed here is that for each $b \in Y$ there is at most one $a \in X$ so that $a \Gamma_f b$.

Putting these observations together, for any function $f: X \to Y$, it is the case that $\Delta_X \subseteq \Gamma_f; f^0$ and $f^0; \Gamma_f \subseteq \Delta_Y$. A valuable principle of set theory is that a relation with these properties determines a function.

**Principle 9: Implicit function definitions**

Suppose $R: X \leftrightarrow Y$ is a relation satisfying

- $\Delta_X \subseteq R; R^0$, and
- $R^0; R \subseteq \Delta_Y$

Then there is a function $f: X \to Y$ so that $R = \Gamma_f$.

Notice that the function determined by a total, determinate relation according to this principle does not seem to be defined by a rule. This is why we call it *implicit* function definition. For any $x \in X$, we have $x \Delta_X x$. So by the first condition there is some $y \in Y$ so that $x R y$ and $y R^0 x$. But if $y'$ also satisfies $x R y'$ and $y' R^0 x$, then $y R^0; R y'$. So the second condition forces $y \Delta_Y y'$. In other words, $y = y'$. So we can define $f(x)$ to be the unique value $y \in Y$ satisfying $x R y$ and $y R^0 x$. We can not necessarily write out an explicit rule $f(x) = \ldots$ to describe this.

### 8.5 Universal relations

The passage from functions to relations that takes $f: X \to Y$ to $\Gamma f: X \leftrightarrow Y$ is *functorial*. This refers to the fact that compositions and identities are preserved: $\Gamma_{g \circ f} = \Gamma_f; \Gamma_g$ and $\Gamma_{id_X} = \Delta_X$. This raises a question of whether there is a similar passage from relations to functions that is somehow the inverse of $\Gamma$.

It turns out that $\Gamma$ is not fully invertible, but there is a systematic way to obtain a function from a relation, illustrating a more general concept of a monad.

A relation $R: U \leftrightarrow X$ is *universal for $X$* if all other relations $S: W \leftrightarrow X$ factor through $R$ via a function graph. That is, for any such $S$ there is a unique function $f: W \to U$ so that $S = \Gamma_f; R$.

If there is a universal relation for $X$, then we can replace...
The symbol $\mathbb{N}$ denotes the set of natural numbers. It is time that we make the structure of $\mathbb{N}$ explicit within our developing ideas about sets and functions.

The structure of $\mathbb{N}$ brings set theory into contact with computation via the crucial idea of recursion. For example, $n!$ (the factorial of $n$) is a function from $\mathbb{N}$ to $\mathbb{N}$ specified recursively by

$$
0! = 1 \\
\text{for } k \geq 1, \quad k! = k \cdot (k-1)!
$$

The axioms for $\mathbb{N}$ are chosen precisely to ensure that a specification like this is guaranteed to define a function uniquely.

9.1 **Sequences and Simple Recurrences**

Let us make the informal word *sequence* official.

**Definition 11: Sequences**

For a set $X$, a **sequence in** $X$ is a function $\sigma : \mathbb{N} \to X$.

For a sequence $\sigma$ in $X$ and natural number $n$, we may write $\sigma_n$ instead of $\sigma(n)$. In any case, the idea is that $\sigma_0, \sigma_1, \sigma_2, \ldots$ are elements of $X$.

As we studied in previous lectures, the basic vocabulary of natural numbers is that (i) there is a starting natural number, $0$, and (ii) for each natural number $k$ there is a next one, $k\succ$. To discuss zero and successor in the language of sets and functions, we stipulate that $0 \in \mathbb{N}$ and that successor is a function $\text{suc} : \mathbb{N} \to \mathbb{N}$.
(given by the rule \( k \mapsto k \cdot \)). The notation \( \text{suc} \) is only needed to bring \( k \cdot \) into the standard function/application notation. So \( \mathbb{N} \) is not just a set: it is a set equipped with a special element and a special function.

Suppose \( X \) is some other similarly equipped set. For example, \( b \in X \) is an element, and \( r: X \to X \) is a function. In other words, \( (X; b, r) \) has structure like \( (\mathbb{N}; 0, \text{suc}) \). So \( X \) may be the natural numbers, but it certainly does not have to be. It has its own \textit{vocabulary} \( b \) is “a beginning” and for each \( x \in X \), the element \( r(x) \) comes “after” \( x \).

Then we should be able define a sequence \( \sigma \) in \( X \), so that \( \sigma_0 = b \), \( \sigma_1 = r(b) \), \( \sigma_2 = r(r(b)) \), \( \sigma_3 = r(r(r(b))) \), and so on.

In general, \( \sigma_k \) should be determined by starting with \( b \) and repeatedly applying \( r \) a total of \( k \) times. In other words, we can summarize \( \sigma \) by two equations:

\[
\begin{align*}
\sigma_0 &= b \\
\sigma_{\text{suc}(k)} &= r(\sigma_k)
\end{align*}
\]

We can make this precise.

**Definition 12: Simple Recurrences and Counting Sets**

A \textit{simple recurrence} is a set equipped with an element \( b \in X \) and function \( r: X \to X \). We will say the \( b \) and \( r \) form a \textit{simple recurrence} on \( X \). To make things easy to read, we will write \( (X; b, r) \) for a simple recurrence on \( X \).

A \textit{universal recurrence} is a simple recurrence \( (\mathbb{N}; z, s) \) so that for any other simple recurrence \( (X; b, r) \) there is exactly one function \( \mathbb{N} \rightarrow X \) so that

\[
\begin{align*}
f(z) &= b \\
f(s(n)) &= r(f(n)) \quad \text{for all } n \in \mathbb{N}
\end{align*}
\]
PRINCIPLE 10: N is a Counting Set

The set \( \mathbb{N} \) of natural numbers with \( 0 \) and \( \text{suc} \) is a universal recurrence.

Consider a simple recurrence \( 1 \overset{b}{\rightarrow} X \overset{r}{\leftarrow} X \). This principle tells us there is a unique function \( f: \mathbb{N} \rightarrow X \) so that

\[
\begin{align*}
  f(0) &= b \\
  f(k^\rightarrow) &= r(f(k))
\end{align*}
\]

In fact, this is just what Postulate 10 tells us: every simple recurrence on a set \( X \) determines a sequence in \( X \).

On the other hand, it is not the case that every sequence is determined by a simple recurrence. Take for example, the sequence \( 0, 1, 0, 2, 0, 3, \ldots \). This cannot be defined (at least not directly) by giving an initial entry and specifying successive entries based only on immediate predecessors.

We need additional techniques for defining more general functions by recursion.

9.2 Primitive Recursion

Evidently, addition, multiplication, factorial, and other familiar functions should be definable using Principle 10. But there are problems to overcome: Addition is not a sequence, and factorial is not definable by a simple recurrence. We need a scheme that generalizes simple recursion to permit (i) dependence on parameters not directly involved in the recursion, and (ii) dependence on \( n \) at each stage of the recursion.

DEFINITION 13:

A **primitive recurrence** in \( X \) with parameters in \( P \) consists of two functions \( P \overset{b}{\rightarrow} X \overset{r}{\leftarrow} P \times \mathbb{N} \times X \).

A **parametric sequence** in \( X \) with parameters in \( P \) is a function \( \alpha: P \times \mathbb{N} \rightarrow X \).

This allows for a very general definition of functions by recursion.
**Postulate 5:** Functions defined by primitive recursion

For any primitive recurrence \( P \overset{b}{\rightarrow} X \leftarrow P \times N \times X \), there is a unique function \( f : P \times N \rightarrow X \) satisfying:

\[
\begin{align*}
    f(p, 0) &= b(p) \\
    f(p, k \rightarrow) &= r(p, k, f(p, k))
\end{align*}
\]

We say that \( f \) is defined by *primitive recursion*.

If the parameter set is trivial (we don’t actually want extra parameters) we can omit them in the notation.

**Example 12:**

The “predecessor” function is defined by the scheme

\[
\begin{align*}
    \text{pred}(0) &= 0 \\
    \text{pred}(n \rightarrow) &= n.
\end{align*}
\]

Addition \( \text{add} : N \times N \rightarrow N \) is defined by

\[
\begin{align*}
    \text{add}(m, 0) &= m \\
    \text{add}(m, k \rightarrow) &= \text{add}(m, k) \rightarrow.
\end{align*}
\]

Of course, we write \( m + n \) instead of \( \text{add}(m, n) \). Notice that, formally, we should give a functions \( b : N \rightarrow N \) and \( r : N \times N \times N \rightarrow N \). Monus (subtraction in \( N \)) can be defined by

\[
\begin{align*}
    m \ominus 0 &= m \rightarrow \\
    m \ominus n \rightarrow &= \text{pred}(m \ominus n)
\end{align*}
\]

Factorial is defined by

\[
\begin{align*}
    0! &= 1 \\
    (n \rightarrow)! &= n\rightarrow \cdot n!
\end{align*}
\]

**Exercises**
72. Define addition \( \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) explicitly by primitive recursion. That is, find functions \( \mathbb{N} \rightarrow \mathbb{N} \) and \( \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) so that

\[
m + 0 = b(m) \\
m + k = r(m, k, m + k)
\]

73. Define multiplication \( \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) by explicit primitive recursion. You may use addition in the definition.

74. For a given function \( f : \mathbb{N} \rightarrow \mathbb{N} \), find a primitive recurrence that defines the function \( \Sigma f : \mathbb{N} \rightarrow \mathbb{N} \) by the informal rule

\[
n \mapsto \sum_{i=0}^{n-1} f(i).
\]

That is, the result will be the sequence \( 0, f(0), f(0) + f(1), f(0) + f(1) + f(2), \ldots \).

75. This is a challenging exercise. Suppose \( f : \mathbb{N} \rightarrow \mathbb{N} \). Define \( \mu_f : \mathbb{N} \rightarrow \mathbb{N} \) so that \( \mu_f(m) \) is the least value \( n \prec m \) for which \( f(n) > 0 \) and is \( m \) otherwise. In other words, \( \mu_f(m) \) will always be a value less than or equal to \( m \). If \( f(n) = 0 \) for all \( n \prec m \), then \( \mu_f(m) = m \). Otherwise \( f(\mu_f(m)) > 0 \) and if \( f(n) > 0 \) for some \( n \prec m \), then \( \mu_f(m) \leq n \).

76. Define by recursion, “halving” as a function from \( \mathbb{N} \) to \( \mathbb{N} \). That is, define \( h : \mathbb{N} \rightarrow \mathbb{N} \) so that \( h(2n) = n \) and \( h(2n + 1) = n \).

9.3 Mutual Recursion and Other Generalizations

Often, we want to define two or more functions simultaneously in terms of each other. For example, consider the following “definitions” of two functions \( f \) and \( g \).

\[
\begin{align*}
f(0) &= 0 \\
g(0) &= 1 \\
f(n \rightarrow) &= f(n) + g(n) \\
g(n \rightarrow) &= f(n)
\end{align*}
\]

So \( f(1) = f(0) + g(0) = 1; f(2) = f(1) + g(1) = f(1) + f(0) = 1 \);
\( f(3) = f(2) + g(2) = f(2) + f(1) = 2 \). In general, \( f \) defines the Fibonacci sequence.

Although the equations defining \( f \) and \( g \) do not follow the format of primitive recursion, they seem to be “safe”. The functions \( f \) and
g are defined mutually: we cannot compute one without the other. Nevertheless, cartesian products allow us to "package" \( f: \mathbb{N} \to \mathbb{N} \) and \( g: \mathbb{N} \to \mathbb{N} \) into a single function \( \langle f, g \rangle: \mathbb{N} \to \mathbb{N} \times \mathbb{N} \). This latter function can be defined by simple recursion. In particular, define \( h: \mathbb{N} \to \mathbb{N} \times \mathbb{N} \) by

\[
\begin{align*}
h(0) &= (0, 1) \\
h(n^\uparrow) &= (\text{pr}(h(n)) + \text{pr}'(h(n)), \text{pr}(h(n)))
\end{align*}
\]

Then it is easily checked that \( f = \text{pr} \circ h \) and \( g = \text{pr}' \circ h \) are the functions we sought. So there was no harm in writing their mutual definition as we did. Indeed, a general scheme for mutually recursive definitions could be worked out.

### 9.4 Why Induction Works

Returning to our original postulates of the natural numbers, let us consider why \( m^\uparrow = n^\uparrow \) should imply \( m = n \). Recall that The predecessor function is defined by primitive recursion:

\[
\begin{align*}
\text{pred}(0) &= 0 \\
\text{pred}(k^\uparrow) &= k
\end{align*}
\]

So if \( m^\uparrow = n^\uparrow \), then \( m = \text{pred}(m^\uparrow) = \text{pred}(n^\uparrow) = n \). In other words, our ability to calculate predecessor by recursion forces the postulate to be true.

Similarly, take a set with two distinct elements. For example, \( \text{BOOL} = \{\text{false, true}\} \). The specific elements are not important, but the fact that \( \text{false} \neq \text{true} \) is. Now define \( \text{is-zero}: \mathbb{N} \to \text{BOOL} \) by

\[
\begin{align*}
\text{is-zero}(0) &= \textbf{true} \\
\text{is-zero}(k^\uparrow) &= \textbf{false}
\end{align*}
\]

Now suppose \( 0 = n^\uparrow \) for some \( n \). Then the recursive definition of \( \text{is-zero} \) would mean that \( \textbf{true} = \text{is-true}(0) = \text{is-true}(n^\uparrow) = \textbf{false} \). That is not possible. So \( 0 \) cannot equal any \( n^\uparrow \).

The Axiom of Induction also follows from our ability to define functions recursively. We can reformulate the Axiom as saying that if \( P \) is a subset of \( \mathbb{N} \) satisfying

- \( 0 \in P \) and
- if \( k \in P \) then \( k \uparrow \in P \) holds for all \( k \in \mathbb{N} \),

then \( P = \mathbb{N} \). We can show that this is true using recursion. It is easier
to reason a bit more generally. Suppose $P$ is a set, and $f: P \to \mathbb{N}$ is a function so that $0 = f(p_0)$ for some $p_0 \in P$. Suppose, moreover, that $t: P \to P$ is a function satisfying $f(t(p)) = f(p) \cdot$ for all $p \in P$. Then we claim that for every $n \in \mathbb{N}$, there is some $p \in P$ so that $f(p) = n$. By simple recursion, there is a function $g: \mathbb{N} \to P$ defined by
\[
g(0) = p_0 \\
g(k \cdot) = t(g(k))
\]
So $f \circ g$ is a function from $\mathbb{N}$ to $\mathbb{N}$. It satisfies the following:
\[
(f \circ g)(0) = f(p_0) \\
= 0 \\
(f \circ g)(k \cdot) = f(t(g(k))) \\
= f(g(k)) \cdot \\
= (f \circ g)(k \cdot)
\]
That is, $f \circ g$ satisfies the same recurrence is the identity on $\mathbb{N}$. Since the primitive recursion postulate requires that any recurrence determines exactly one function, it must be the case that $id_\mathbb{N} = f \circ g$. So for any $n \in \mathbb{N}$, $f(g(n)) = n$.

Now take $f$ to be an inclusion map $incl_P: P \to \mathbb{N}$ where $P$ is a subset of $\mathbb{N}$. The conditions on $f$ translate to saying that $0 \in P$ and that $P$ is closed under taking successors. So for every $n \in \mathbb{N}$, it must be the case that $n \in P$. Hence the two are equal.

In summary, Postulate 5 says that (primitive) recursion is a permissible way to define functions involving natural numbers. This leads to proofs that our original natural number postulates (from Chapter ??) are forced to be true. In effect, they are redundant now. This reinforces the idea that computation (in the form of recursion) is foundational for mathematics.
Natural numbers constitute an important example of a general process whereby “bigger” objects are built up from “smaller” ones. The Axiom of Induction captures the idea of building up, providing a method for proving facts about natural numbers. In this chapter, we develop an analogous way to think about lists.

10.1 List Basics

In this section, we concentrate on the fundamental concept of lists. The idea is really meant to be the familiar one, so a list of “to do” items is a list. The alphabetized names on a class roster is a list. We will write lists using square brackets, following notation in many common programming languages. So for example, \([2, 3, 5, 7]\) is the list of the prime numbers less than 10 in ascending order. For lists, we expect the order to matter. So \([7, 5, 3, 2]\) is not the same as \([2, 3, 5, 7]\).

Like natural numbers, lists can be built up by starting with an empty list and incrementally adding items. Though we have choices for how to formalize this, we will follow a standard that has developed in computer science.

Given the list \([2, 5, 1]\), we will say that 2 is the head and that \([5, 1]\) is the tail. So “head” refers to the initial item and “tail” refers to the shorter list. We may build another list with head 0, and tail \([2, 5, 1]\), resulting in \([0, 2, 5, 1]\).

Following the notation in the language Haskell, we denote the operation of prepending an item to a list by `::`. So \(0 :: [2, 5, 1]\) is the list \([0, 2, 5, 1]\).

An empty list, having no head or tail, is also possible. It is denoted by `[]`. The empty list, together with prepending items, gives us a way to construct any list we want.
Example 13:

Here are some examples.

• $5 :: 6 :: [4, 5]$ is the same as $5 :: [6, 4, 5]$, which is the same as $[5, 6, 4, 5]$.

• $[]$ is the empty list

• $1 :: []$ is the same as $[1]$

• $1 :: 2 :: 3 :: 4 :: []$ is the same as $[1, 2, 3, 4]$.

• $1 :: 2$ is nonsense – we can’t prepend an item to something that is not a list.

Now we have a decision to make. Is it reasonable to form a list that consists of different kinds of data? The examples so far all the items in a list have been natural numbers. But would it make sense to make a list $[1, \sin]$?

Vocabulary 6: Basic Vocabulary of Lists

• There is a special list, which we call the empty list and denote by $[]$.

• For any thing $x$ and any list $L$, there is another list, having $x$ as the head and $L$ as the tail. We denote the result as $x :: L$.

To discuss later, lists do not form a set. So we will never write something list $L \in \text{List}$. That notation simple doesn’t make sense.

A thing appearing in a list is called an item of the list. Specifically, $[]$ has no items. A list $x :: L$ has $x$ as its initial item. Any item of $L$ is an item of $x :: L$.

We abbreviate lists using square brackets. So for example, $[2, 3]$ is an abbreviation for $2 :: 3 :: []$. When we write a series of :: operations like this, they associate right to left. So $2 :: 3 :: []$ means $2 :: (3 :: [])$.

A list may consist of items that are all similar. For example, the list in the above description is a list of natural numbers. But lists are not required to be homogeneous. For example, $[4, [3, 4]]$ is a list consisting of two items. The first is the natural number $4$. The second is the list $[3, 4]$. So $[4, [3, 4]]$ is an example of a heterogeneous list.

We will need to think separately about homogeneous lists later. For now, we do not make any assumptions about items.
As with the natural numbers, we need to think about axioms that prevent strange behavior. These are closely analogous to the axioms of natural numbers.

**Postulate 6:** Empty list has no head or tail.

For any thing \( x \) and any list \( L \), \([\ ] \neq x :: L\).

Likewise, a list that is not empty can only be built one way.

**Postulate 7:** Non-empty lists are determined by head and tail

For any things \( x \) and \( y \) and lists \( L \) and \( M \), if \( x :: L = y :: M \), then \( x = y \) and \( L = M \).

For example, if I tell you \([2,3,4,5] = x :: L\), then you know immediately that \( x \) must equal 2 and \( L \) must equal \([3,4,5]\). No other interpretation is possible.

Lists need an induction axiom that ensures that all lists are built up from \([\ ]\).

**Postulate 8:** The Axiom of List Induction

No lists can be removed without violating Vocabulary 6.

This axiom justifies proofs about lists using a scheme almost identical to simple arithmetic induction.
**List induction**

To prove some property is true about all lists,

*Basis:* Prove that the property is true for `[]`.

*Inductive hypothesis:* Assume that the property is true for some list `K`.

*Inductive step:* Prove that for any thing `x`, the property is true for `x :: K`.

Conclude that the property is true for all lists.

Operations on lists can also be defined by schemes similar to addition and multiplication on natural numbers. For example, every list has a length. Writing `len(L)` for the length of a list, `len([2, 3, 4]) = 3`. A precise definition is now easy to formulate.

**Algorithm 10: Length of a list**

For a list `L`, the *length* of `L`, denoted by `len(L)`, is the natural number defined by

\[
\begin{align*}
len([]) &= 0 \\
len(x :: L) &= len(L) \cdot x \\
\end{align*}
\]

**Example 14:**

\[
\begin{align*}
len([2, 3, 4]) &= len(2 :: [3, 4]) \\
&= len([3, 4]) \cdot 2 \\
&= len(3 :: [4]) \cdot 2 \\
&= len([4]) \cdot 3 \\
&= len([4]) \cdot 3 \\
&= 0 \cdot 3 \\
&= 3
\end{align*}
\]
Another common operation on lists is *concatenation*:

\[ [2, 3, 4] + [4, 1, 3] = [2, 3, 4, 1, 3] \]

, whereby the two lists are glued together in their original orders. This is defined precisely by the following.

**Algorithm 11: List concatenation**

For lists \( L \) and \( M \), their *concatenation*, denoted by \( L + M \), is a list. For all lists \( M \), the following are true.

\[
\begin{align*}
[] + M &= M \\
(x : K) + M &= x : (K + M)
\end{align*}
\]

for any thing \( x \) and any list \( K \)

Notice that concatenation and addition are defined essentially the same way: 0 and \( [] \) play the same roles, \( n \cdot \) and \( x : L \) play the same roles.

**Example 15: List concatenation**

To calculate \( [4, [5, 2], 1] + [3, 4, 1] \), we follow the algorithm:

\[
\begin{align*}
&= 4 : (([5, 2] : 1 : []) + [3, 4, 1]) \\
&= 4 : [5, 2] : 1 : ([3, 4, 1]) \\
&= 4 : [5, 2] : 1 : [3, 4, 1] \\
&= [4, 5, 2, 1, 3, 4, 1]
\end{align*}
\]

[[Abbreviation]

Now we can prove some useful facts about lists.
**Proposition 8:** \([\hat{\ }]\) is the identity for +

For list \(L\),
\[
[\hat{\ }] + L = L
\]
and
\[
L = L + [\hat{\ }].
\]

**Proof:** By definition \([\hat{\ }] + L = L\) is always true. We proceed by induction on \(L\) to prove the second equality. The proof should look familiar (see the proof of Lemma 2).

**Basis:** \([\hat{\ }] + [] = []\) is true by definition of +.

**Inductive hypothesis:** Assume \(K + [] = K\) for some list \(K\).

**Inductive step:** Suppose \(x\) is some thing. We need to show that \((x : K) + [] = x : K\).

\[
(x : K) + [] = x : (K + []) \quad \text{[by definition of +]}
= x : K \quad \text{[by the Inductive Hypothesis]}
\]

Thus (by the Axiom of List Induction), \(L + [] = L\) holds for all lists. \(\square\)

---

**Proposition 9:** Concatenation is associative

For all lists \(L\), \(M\) and \(N\),
\[
L + (M + N) = (L + M) + N.
\]

**Proof:** We prove this by induction on \(L\). Again, this should look familiar. It is almost identical to the proofs that addition and multiplication are associative.

Suppose \(M\) and \(N\) are fixed lists.

**Basis:** The goal is to show that \([\hat{\ }] + (M + N) = ([\hat{\ }] + M) + N\). This follows easily from the definition of +.

**Inductive hypothesis:** Suppose \(K + (M + N) = (K + M) + N\) for some list \(K\).
Inductive step: The goal is to show that for any \( x, (x :: K) + (M + N) = ((x :: K) + M) + N \). For any thing \( x, \)

\[
(x :: K) + (M + N) = x :: (K + (M + N)) \\
= x :: ((K + M) + N) \quad \text{Inductive Hypothesis} \\
= (x :: (K + M)) + N \\
= ((x :: K) + M) + N \quad \text{Def. of +}
\]

So \( L + (M + N) = (L + M) + N \) is true for all \( L \). Since the proof does not depend on any special properties of \( M \) and \( N \) (except that they are both lists), the result is true for all lists \( M \) and \( N \). \( \square \)

Concatenation is analogous to addition in the following way as well.

**Proposition 10:** Concatenation is cancellative

For lists \( L, M \) and \( N, \)

- \( L + M = L + N \) implies \( M = N \); and
- \( L + N = M + N \) implies \( L = M \).

**Proof:** Exercise. \( \square \)

Here is another useful fact that we can prove by induction relating length to concatenation.

**Proposition 11:** \( len \) is a homomorphism

\[ \text{For any lists } L \text{ and } M, len(L + M) = len(L) + len(M). \]

**Proof:** The proof is by induction on \( L \). Suppose \( M \) is some fixed list.

**Basis:** The goal is to show that \( len([]) + len(M) = len([] + M) \). But \( len([]) + len(M) = 0 + len(M) = len(M) = len([] + M) \) by definition of + and the fact that 0 is the additive identity.

**Inductive Hypothesis:** Suppose \( len(K + M) = len(K) + len(M) \) holds for some particular list \( K \).

**Inductive Step:** The goal is to show that for any \( x, \)

\[ len((x :: K) + M) = len((x :: K)) + len(M). \]
Suppose x is some thing.

\[ \text{len}(x:K + M) = \text{len}(x:(K + M)) \]
\[ = \text{len}(K + M)^\wedge \text{Def. of +} \]
\[ = (\text{len}(K) + \text{len}(M))^\wedge \text{Def. of len} \]
\[ = \text{len}(K)^\wedge + \text{len}(M) \text{ Inductive Hypothesis} \]
\[ = \text{len}(x:K) + \text{len}(M) \text{ Lemma 1} \]

□

Often we use a list somewhat informally without all the punctuation. For example, we might say “Consider a list \( a_0, a_1, \ldots, a_{n-1} \) of real numbers.” If we do not intend to use the list itself for anything special, but only want to think about the numbers \( a_0 \) through \( a_n \), then there is no need to be formal about it. Also, there is no harm in writing something like this: \( a_5, a_6, a_7, a_8 \), where the indices start at 5. The default is to start at 0, but that is merely a convention.

Sometimes, it is even useful to avoid punctuation completely. For example, in text processing words are typically regarded as being lists of letters: the word marmoset is the list \([m, a, r, m, o, s, e, t]\).

Exercises

77. Prove Proposition 10.

78. Show that concatenation is not commutative.

79. Write a recursive algorithm for list reversal. That is, define an operation \( \text{rev} \) on lists so that, for example, \( \text{rev}([2, 3, 4]) = [4, 3, 2] \).

[Hint: Just ask yourself what is the reverse of the empty list, and what is the reverse of \( x : L \).]

10.2 List Itemization

In a list \( L \), the items are in order. So we can refer to items by their position in the list. There are two standards in mathematics for doing this. Either we start counting from 1 or from 0. Although starting from 0 (meaning that the initial item of a list is item number 0) may take some getting used to, it actually makes many calculations simpler. For that reason, most programming languages use this convention for a lists and arrays. I will consistently start with 0.
**Algorithm 12**: Items in a list.

Suppose L is a list and i is a natural number so that i < len(L). Note that len([]) = 0, so there is no i < len([]). We define L_i as follows.

- []_i is never defined because i ≠ len([])
- (x :: L)₀ = x
- (x :: L)ₖ₋₁ = Lₖ provided that Lₖ is defined

This is a precise way of explaining that in a list, for example L = [a, b, c, d, e], we can refer to an item by its index, so that L₀ = a, L₁ = b and so on, up to L₄ = e. Notice that, for this example, L₅, L₆ and so on are undefined.

**Example 16**:

Suppose L = [a, b, c, d, e]. We can calculate L₃ explicitly step by step.

\[
L₃ = [a, b, c, d, e]₃ \\
= (a :: b :: c :: d :: e :: [])₀ ~ ~ ~ \\
= (b :: c :: d :: e :: [])₀ ~ ~ \\
= (c :: d :: e :: [])₀ ~ ~ \\
= (d :: e :: [])₀ \\
= d
\]

Of course, this is just a very careful way to find item number 3 in the list. In every day use, we humans would not do this. We would simply count forward from the beginning of the list. This definition explains precisely what "simply count forward" means.

**Exercises**

80. Suppose L is a list and i < len(L). Then it makes sense to think about the list in which L_i is removed. For example, for L = [a, b, c], removing L₁ results in the list [a, c]. Let us denote the result of removing item i from list L as L \ i. So [a, b, c] \ l = [a, c].
In this exercise, you define \( L \setminus i \) explicitly. Explain your answers for each point.

(a) Should \([ \ ] \setminus 0 \) be defined? If not, why not? If so, what should it be?
(b) What should \((x :: L) \setminus 0 \) be?
(c) What should \((x :: L) \setminus k^\sim \) be? When should it be defined and undefined?
(d) Now write your definition for \( L \setminus i \) in a layout similar to Definition 12.

81. Suppose \( L = [3, 2, 3, 3, 5] \) and \( M = [0, 1, 2, 3, 4, 5] \). Calculate the following explicitly step by step.

(a) \( \text{len}(L) \)
(b) \( L_4 \)
(c) \( (L + M)_9 \)
(d) \( L \setminus 4 \).
(e) \( ((M + L) \setminus 3)_5 \)

10.3 Slices

Suppose \( L = [a_0, \ldots, a_{n-1}] \) is a list of length \( n \) and that \( i \leq j \leq n \). We can extract the sublist of \( L \) consisting of items indexed \( i, i+1, \ldots, j-1 \), writing \( L[i : j] \) for the list \([a_i, a_{i+1}, \ldots, a_{j-1}]\). The reader may think this is strange usage because \( L[i : j] \) stops just one item shy of \( j \). But it is exactly the same notation used, for example, in Python for what are called “slices” of a list. One reason this notation makes sense is that it works well with concatenation. Namely, if \( i \leq j \leq k \leq n \), then \( L[i : j] + L[j, k] = L[i, k] \).

So here is a formal definition of slicing.
ALGORITHM 13: List slicing

For a list $L$ and natural numbers $i \leq j \leq \text{len}(L)$, $L[i : j]$ is a list defined by the following:

\[
L[0 : 0] = [] \\
(x :: L)[0 : j] = x :: (L[0 : j]) \\
(x :: L)[i : j] = L[i : j]
\]

Exercises


83. Show that for any list $L$ and any $i \leq \text{len}(L)$, it is the case that $L[i : i] = []$.

84. Show that for any list $L$ and any $i \leq j \leq k \leq \text{len}(L)$, it is the case that $L[i : j] + L[j : k] = L[i : k]$. 
Part III
Applications
11

Minimum and Maximum

The ordering of natural numbers gives rise to two other operations: min and max. The minimum of two natural numbers \( m \) and \( n \) is, of course, the smaller of the two. It makes sense to say this because \( \leq \) is linear. That is, either \( m \leq n \) — in which case \( m \) is the minimum — or \( n < m \) — in which case \( n \) is the minimum. We write \( \min(m, n) \) for this. The maximum is written as \( \max(m, n) \). Both of these can be specified by algorithms.

**Algorithm 14: Minimum**

For natural numbers \( m \) and \( n \), \( \min(m, n) \) is calculated by

\[
\begin{align*}
\min(0, k) &= 0 \\
\min(j \rightarrow, 0) &= 0 \\
\min(j \rightarrow, k \rightarrow) &= \min(j, k) \\
\end{align*}
\]

**Algorithm 15: Maximum**

For natural numbers \( m \) and \( n \), \( \max(m, n) \) is calculated by

\[
\begin{align*}
\max(0, k) &= k \\
\max(j \rightarrow, 0) &= j \\
\max(j \rightarrow, k \rightarrow) &= \max(j, k) \\
\end{align*}
\]

Exercises
85. Calculate explicitly \( \min(3, 6) \).

86. Calculate explicitly \( \max(4, 3) \).

The important property of \( \min \) and \( \max \) is summarized by the following lemma.

**Lemma 17: Characterizing \( \min \) and \( \max \)**

For natural numbers \( m, n \) and \( p \),

\[
\begin{align*}
p &\leq \min(m, n) \iff p \leq m \text{ and } p \leq n \\
\max(m, n) &\leq p \iff m \leq p \text{ and } n \leq p.
\end{align*}
\]

**Proof:** We look only at \( \min \) because the proof for \( \max \) is essentially identical.

By cases, either \( m \leq n \) or \( n < m \). In the first case, \( \min(m, n) = m \), so \( \min(m, n) \leq m \) by reflexivity, and \( \min(m, n) \leq n \) because \( m \leq n \).

In the latter case, \( \min(m, n) = n \) by reflexivity, and \( \min(m, n) \leq m \) because we have assumed \( n < m \).

Suppose \( p \leq m \) and \( p \leq n \). By linearity, either \( m \leq n \) or \( n < m \). In either case, \( p \leq \min(m, n) \). \( \square \)

It is worth noting that \( \min(m, n) \) is actually the unique natural number satisfying the two conditions of the lemma. That is, suppose (i) \( q \leq m, q \leq n \) and (ii) for all natural numbers \( p \), if \( p \leq m \) and \( p \leq n \) then \( p \leq q \). Then by (i), \( q \leq \min(m, n) \). By (ii), \( \min(m, n) \leq q \). So \( q = \min(m, n) \). This tells us that \( \min(m, n) \) is, in a sense, optimal for solving the two inequalities \( x \leq m \) and \( x \leq n \) simultaneously. We summarize the lemma by saying that \( \min(m, n) \) is the greatest lower bound of \( m \) and \( n \). Likewise, \( \max(m, n) \) is the least upper bound.

Some simple facts about \( \min \) and \( \max \) derive directly from these characterizations plus some facts about addition.

In particular, together with addition, \( \min \) and \( \max \) make the natural numbers into something called a distributive lattice ordered monoid. Basically, this means that addition together with \( \min \) and \( \max \) cooperate in specific ways that augment the basic laws of arithmetic. The most useful laws having to do with \( \min \) and \( \max \) follow.
Basic Laws of min and max

For any natural numbers, m, n and p:

Characterization via \( \leq \)
- \( m \leq \min(n, p) \) if and only if \( m \leq n \) and \( m \leq p \)
- \( \max(m, n) \leq p \) if and only if \( m \leq p \) and \( n \leq p \)

Associativity
- \( \min(m, \min(n, p)) = \min(\min(m, n), p) \)
- \( \max(m, \max(n, p)) = \max(\max(m, n), p) \)

Commutativity
- \( \min(m, n) = \min(n, m) \)
- \( \max(m, n) = \max(n, m) \)

Idempotency
- \( \min(m, m) = m \)
- \( \max(m, m) = m \)

Absorptivity
- \( m = \min(m, \max(n, m)) \)
- \( m = \max(m, \min(n, m)) \)

Distributivity
- \( m + \min(n, p) = \min(m + n, m + p) \)
- \( m + \max(n, p) = \max(m + n, m + p) \)
- \( \max(m, \min(n, p)) = \min(\max(m, n), \min(m, p)) \)
- \( \min(m, \max(n, p)) = \max(\min(m, n), \min(m, p)) \)
- \( m \cdot \min(n, p) = \min(m \cdot n, m \cdot p) \)
- \( m \cdot \max(n, p) = \max(m \cdot n, m \cdot p) \)

Modularity
- \( m + n = \min(m, n) + \max(m, n) \)

We will not prove most of these as they follow easily from arithmetic laws. Nevertheless, a sampling follows.

**Lemma 18:** Addition distributes over min

\[ m + \min(n, p) = \min(m + n, m + p) \] for all natural numbers \( m, n, p. \)

**Proof:** Because \( \min(n, p) \leq n \) and addition is monotonic, \( m + \min(n, p) \leq m + n. \) Likewise \( m + \min(n, p) \leq m + n. \) So \( m + \min(n, p) \leq \min(m + n, m + p). \)

So to complete the proof, we must show that \( \min(m + n, m + p) \leq m + \min(n, p). \) Suppose \( k \leq \min(m + n, m + p). \) Then \( k \leq m + n \) and \( k \leq m + p. \) If \( k \leq m \) then \( k \leq m + \min(n, p) \) obviously. Otherwise, \( m < k \) by Linearity. So \( m + d = k \) for some \( d. \) Hence \( m + d = m + n \)

Like the basic laws of arithmetic presented in Chapter 1, there laws are organized here to emphasize similarities between min and max. Pay attention to that.

This is a good illustration of why antisymmetry of \( \leq \) is useful. We wish to prove \( m + \min(n, p) = \min(m + n, m + p) \). Instead, we prove \( m + \min(n, p) \leq \min(m + n, m + p), \) and separately, \( \min(m + n, m + p) \leq m + \min(n, p). \) Then we let antisymmetry take us the rest of the way.
and \( m + d \leq m + p \). Since \( \leq \) is order reflecting, \( d \leq \min(n, p) \).
Consequently, \( k = m + d \leq m + \min(n, p) \). We have thus shown that
\( k \leq \min(m + n, m + p) \) implies \( k \leq m + \min(n, p) \). In particular, this
applies to \( \min(m + n, m + p) \). \( \square \)

Exercises

87. Calculate the following values. Show work.

(a) \( \min(5, \min(4, 6)) \)
(b) \( \min(5, \max(4, 6)) \)
(c) \( \min(340, \max(234, 340)) \)
(d) \( \min(5, \max(3, \min(\max(1, 2), 7))) \)
(e) \( \min(5 + \max(4 + \min(3 + \max(7, 8), 3 + \min(7, 8)), 6), 7) \)

88. Prove that \( \min \) distributes over \( \max \).

89. Prove that multiplication distributes over \( \min \).
Divisibility: Ordering the Natural Numbers by Multiplication

Divisibility of natural numbers is a relation defined by analogy with the standard order. In the standard order, \( m \leq n \) means that \( m + d = n \) for some \( d \). Because multiplication satisfies many of the same laws as addition (it is commutative, associative, etc.), a similar definition is possible in terms of multiplication.

**Definition 14: Divisibility**

For natural numbers \( m \) and \( n \), say that \( m \) divides \( n \) if and only if \( m \cdot q = n \) for some natural number \( q \). We write \( m \mid n \) when \( m \) divides \( n \), and write \( m \notmid n \) when \( m \) does not divide \( n \).

An alternative way to say that \( m \) divides \( n \) is \( n \) is a multiple of \( m \).

The distinction between \( m \leq n \) and \( m \mid n \) is precisely that the former is defined by addition and the latter by multiplication. So we can transfer many facts about \( \leq \) to facts about \( \mid \) simply by noticing that they depend on analogous laws of arithmetic.

For example, the relation \( \leq \) is reflexive — meaning that \( m \leq m \) for all \( m \) — *because* \( 0 \) is the identity for addition. The relation \( \mid \) is reflexive similarly because \( 1 \) is the identity for multiplication. That is \( m \mid m \) because \( m \cdot 1 = m \). Likewise, \( \leq \) is transitive because addition is associative; \( \mid \) is transitive because multiplication is associative.

Divisibility is also anti-symmetric — meaning that \( m \mid n \) and \( n \mid m \) implies that \( m = n \). But a proof of this hints at an important difference between \( \mid \) and \( \leq \).

Recall that we proved that \( m \leq n \) and \( n \leq m \) implies \( m = n \) using...
cancellativity of addition. But multiplication is only cancellative for non-zeros. That is, \( m \cdot p = n \cdot p \) implies \( m = n \) only when \( p > 0 \). So we need to treat 0 as a special case.

**LEMMA 19:** Divisibility is anti-symmetric

For any natural numbers \( m \) and \( n \) if \( m \mid n \) and \( n \mid m \), then \( m = n \).

**Proof:** Suppose \( m \) and \( n \) are natural numbers satisfying \( m \mid n \) and \( n \mid m \). Thus, there are natural numbers \( q \) and \( r \) so that

\[
\begin{align*}
m \cdot q &= n \\
n \cdot r &= m
\end{align*}
\]

Suppose \( m = 0 \). Then by the first equation \( n = 0 \) as well. Suppose \( m = p \wedge \) for some \( p \). Then

\[
p \wedge \cdot q \cdot r = p \wedge
\]

So cancelling \( p \wedge \) yields \( q \cdot r = 1 \). But natural numbers also satisfy the Law of Integrality. So \( q = 1 \). \( \Box \)

The divisibility relation begins to be more interesting when we realize that it is not linear. For example, 4 does not divide 13 and 13 does not divide 4. Apparently, the structure of the natural numbers with respect to \( \mid \) is much more complicated than with respect to \( \leq \).

Note that 1 \( \mid m \) is true for any \( m \), simply because 1 \( \cdot m = m \). And \( m \mid 0 \) is true for any \( m \) because \( m \cdot 0 = 0 \). So 1 is “at the bottom” of the divisibility relation and 0 is “at the top”.

Figure 12.1 shows a fragment of the natural numbers with respect to divisibility.

It may seem strange to claim that 0 \( \mid 0 \). Some people find it so irritating that they simply rule 0 out of consideration, and declare that 0 \( \mid 0 \) is undefined. Technically, this means they are committed to saying \( m \mid n \) if and only if there is a unique \( q \) so that \( m \cdot q = n \). Since 0 \( \cdot q = 0 \) for all \( q \), there is no unique value. This is fine, but I prefer to understand 0 \( \mid 0 \) just to mean 0 \( \cdot q = 0 \) for some \( q \).
For each of the following pairs \((m, n)\) of natural numbers, determine whether or not \(m \mid n\).

90. \((4, 202)\)
91. \((7, 49)\)
92. \((11, 1232)\)
93. \((9, 19384394)\)
94. \((n, 6n)\)
95. \((26, 65)\)

Before closing this section, we note additional facts about how divisibility interacts with arithmetic.
Lemma 20: Technical facts about divisibility

1. \( m \mid n \) implies \( m \mid n \cdot p \)

2. If \( m \mid n \), then \( m \mid p \) if and only if \( m \mid (n + p) \)

3. If \( n < m \) and \( m \mid (pm + n) \), then \( n = 0 \).

4. \( m \mid n \) and \( n > 0 \) implies \( m \leq n \).

Proof: When \( m = 0 \), all of these are trivial, as \( 0 \mid n \) holds if and only if \( n = 0 \). So suppose \( m \) is positive.

1. is due associativity.

2. is due to distributivity.

3. Suppose \( n < m \) and \( md = mp + n \). If we can show that \( d = p \), then it follows that \( n = 0 \). First, note that \( mp \leq md \). And since \( m \) is positive, it can be cancelled. So \( p \leq d \). Second, \( md = mp + n < mp + m = m(p \nrightarrow) \). Again, cancelling \( m \) yields \( d < p \nrightarrow \). So \( d \leq p \).

4. follows from (3).

\( \square \)

Exercises

96. Fill in the details of the proofs of Lemma 20:1,2 and 4.

12.1 Quotients and Remainders

Without rational numbers, division still makes sense, provided we account for the fact that sometimes numbers don’t divide evenly. When you first learned about division, you learned about quotients
and remainders. For example, you may have calculated a long division,

\[
\begin{array}{c|c}
925 & 9 \\
13 & \\
\hline
12034 \\
11700 \\
\hline
334 \\
260 \\
\hline
74 \\
65 \\
\hline
9 \\
\end{array}
\]

showing that the quotient is 925 with remainder 9. In any correct long division \( m \div n \), the remainder (call it \( r \)) must be less than \( n \). In the example, \( 9 < 13 \). The quotient (call it \( q \)) and \( r \) together satisfy \( m = n \cdot q + r \). In the example, \( 12034 = 13 \cdot 925 + 9 \).

So the long division calculations you performed in elementary school were really an implementation of the following fact.

---

**Theorem 2: Natural number division**

For any natural number \( m \) and any positive natural number \( n \), there is a unique pair of natural numbers \( q \) and \( r \) satisfying

1. \( m = n \cdot q + r \) and
2. \( r < n \).

**Proof:** First, we prove that if \( q \) and \( r \) satisfy the stated conditions, and so do \( q' \) and \( r' \), then \( q = q' \) and \( r = r' \). This will prove that there can be at most one pair of natural numbers satisfying the stated conditions.

Suppose \( n \cdot q + r = n \cdot q' + r' \) and \( r < n \) and \( r' < n \). If \( r < r' \), then there is some \( e \) so that \( r + e < r' \). Hence \( q \cdot n = q' \cdot n + e \). But then \( n \mid n \cdot q' + e \). But \( e \) is strictly less than \( n \), so this is impossible. Thus it can not be the case that \( r < r' \). For the same reason, it can not be the case that \( r' < r \). So \( r = r' \). By cancellativity of addition, \( q \cdot n = q' \cdot n \). And since \( n \) is positive, cancellativity of multiplication yields \( q = q' \).

Now we prove that there actually is a pair of numbers \( q \) and \( r \) satisfying the conditions. For this, we proceed by induction on \( m \). Fix a positive natural number \( n \).

- **[Basis]** The goal is to find \( q \) and \( r \) so that \( 0 = q \cdot n + r \) and \( r < n \).

  Clearly, \( q = 0 \) and \( r = 0 \) do the job.
**Divisibility: Ordering the Natural Numbers by Multiplication**

- **[Inductive hypothesis]** Assume that for some \( k \in \mathbb{N} \), there are natural numbers \( p \) and \( s \) satisfying \( k = p \cdot n + s \) and \( s < n \).

- **[Inductive Step]** The goal is to find \( q \) and \( r \) so that \( k \cdot n = q \cdot n + r \) and \( r < n \). By the inductive hypothesis, \( k \cdot n = p \cdot n + s \cdot n \). There are two cases to consider: either \( s \cdot n < n \), or \( s \cdot n = n \) — \( s \cdot n \) can not be strictly greater than \( n \) because \( s < n \). Suppose \( s \cdot n < n \). Then let \( q = p \) and let \( r = s \cdot n \). Suppose \( s \cdot n = n \). Then \( k \cdot n = p \cdot n + n = p \cdot n + 0 \). So let \( q = p \cdot n \) and \( r = 0 \).

So the claim is true for all \( m \in \mathbb{N} \) and all positive \( n \in \mathbb{N} \). □

This theorem indicates that division of natural numbers works, as long as we account for both quotient and remainder. It is useful to have notation for these.

**Definition 15:**

For any natural number \( m \) and positive natural number \( n \), let \( m \div n \) denote the *quotient* and \( m \mod n \) the *remainder* of dividing \( m \) by \( n \). That is, \( m \div n \) and \( m \mod n \) are the unique natural numbers so that

1. \( m = n \cdot (m \div n) + (m \mod n) \) and
2. \( (m \mod n) < n \).

The notation \( \div \) is borrowed from the programming language Python. It is intended to avoid confusion with real number division \( x/y \). Python, Java, C and many other languages use a percent sign for remainder, but unfortunately, its precise meaning differs from language to language. The notation mod avoids this ambiguity.

Notice that \( \div \) can be characterized in relation to \( \leq \). First, evidently if \( m \leq n \) and \( p > 0 \), then \( m \div p \leq n \div p \). A proof of this is a simple exercise. So division by a positive \( p \) is monotonic: \( m \leq n \) implies \( p \cdot m \leq p \cdot n \).
Lemma 21: Division is adjoint to multiplication

For natural numbers \( m, n \) and \( p \), with \( p \) positive, it is the case that \( p \cdot m \leq n \) if and only if \( m \leq n // p \).

**Proof:** Suppose \( m \leq n // p \). Then \( p \cdot m \leq p \cdot (n // p) \) because multiplication by \( p \) is monotonic. Moreover, \( p \cdot (n // p) \) + (\( n \) mod \( p \)) = \( n \). So \( p \cdot m \leq n \).

Conversely, suppose \( p \cdot m \leq n \). Then \( (p \cdot m) // p \leq n // p \). But \( m = (p \cdot m) // p \). □

The following shows that a dual version of division is also possible. That is, for \( m \in \mathbb{N} \) and positive \( n \in \mathbb{N} \), there is a natural number \( s \) so that \( s \leq p \) is and only if \( m \leq n \cdot p \).

**Corollary 1:** Natural number division, dual version

For any natural number \( m \) and any positive natural number \( p \), there is a unique pair of natural numbers \( s \) and \( t \) satisfying

1. \( m + t = p \cdot s \) and
2. \( t < p \).

Moreover, for any natural number \( n, s \leq n \) if and only if \( m \leq n \cdot p \).

**Proof:** By division, there are natural numbers \( q \) and \( r \) so that \( m = n \cdot q + r \) and \( r < n \). If \( r = 0 \), then \( m + r = n \cdot q \). Otherwise, \( r < n \) and there is a “difference” \( t \) so that \( r + t = n \). Clearly, \( t < n \) because \( r > 0 \). So \( m + t = n \cdot q + r + t = n \cdot q + n = n \cdot (q + 1) \).

Suppose \( s \leq p \). Then \( m \leq n \cdot s \leq n \cdot p \). Conversely, if \( m \leq n \cdot p \), then

\[
 n \cdot s = m + t < n \cdot p + n = n \cdot (p + 1).
\]

But multiplication by \( n \) is order reflecting, so \( s < p + 1 \). That is, \( s \leq p \). □

There is no agreed-upon notation for the dual version of division mainly because it simply is not used much. Though it is tempting to write \( p \\backslash\backslash m \), we do not need to introduce a new bit of notation.

Exercises
97. Calculate the following:

(a) \(24 \div 7\)
(b) \(1000000 \mod 1000001\)
(c) \(13 \mod 8\)
(d) \(8 \mod 5\)
(e) \(5 \mod 3\)
(f) \(3 \mod 2\)
(g) \(2 \mod 1\)

98. Show that for any \(m \in \mathbb{N}\), any positive \(n \in \mathbb{N}\) and \(p \in \mathbb{N}\), it is the case that

\[(p \cdot m) \div (p \cdot n) = m \div n\]

and

\[(p \cdot m) \mod (p \cdot n) = p \cdot (m \mod n).\]
Recall that the minimum of two natural numbers — written $\min(m, n)$ — is characterized as the greatest lower bound of $m$ and $n$ with respect to the standard order $\leq$. Likewise, $\max(m, n)$ is the least upper bound. That is, for any $m$, $n$ and $p$, they satisfy

\[ p \leq \min(m, n) \iff p \leq m \text{ and } p \leq n \]
\[ \max(m, n) \leq p \iff m \leq p \text{ and } n \leq p \]

This chapter concerns analogous operations with respect to divisibility. The idea is simple. If any two natural numbers have a greatest lower bound and a least upper bound with respect to the “natural” ordering of $\leq$, then perhaps they also have a greatest lower bound and a least upper bound with respect to divisibility.

In fact, they do. The greatest lower bound with respect to divisibility is called the greatest common divisor and is denoted $\gcd(m, n)$.

The least upper bound with respect to divisibility is called the least common multiple and is denoted by $\text{lcm}(m, n)$.

Though it may not be immediately obvious that greatest common divisor or least common multiple for any two natural numbers exists, we have a good inkling that they do.

When you simplify a fraction, say $\frac{24}{18}$, you remove any common factors. For example, $\frac{24}{18} = \frac{8 \cdot 3}{6 \cdot 3}$. So the simplified fraction is $\frac{4}{3}$. You could have factored out $\frac{2}{3}$ or $\frac{3}{3}$, but $\frac{8}{6}$ is in some sense the best you can do. It divides 24 and 18, and is divisible by any other common divisor. In short 6 is the greatest common divisor of 24 and 18. You have performed the operation of finding greatest common divisors ever since you learned to simplify fractions. So now our task is simply to put that performance on a solid footing.
We do this by showing that for every two natural numbers $m$ and $n$, natural numbers $\text{gcd}(m, n)$ and $\text{lcm}(m, n)$ exist so that

$$p \mid \text{gcd}(m, n) \iff p \mid m \text{ and } p \mid n$$
$$\text{lcm}(m, n) \mid p \iff m \mid p \text{ and } n \mid p$$

Compare these two conditions to the characterizing conditions for min and max. They are identical, except that $\leq$ is replaced by $\mid$.

A proof that every two natural numbers have a greatest common divisor is quite old. Euclid included one proof in his geometry text. The proof given here is based on a proof due to the 18th century mathematician Bezout. This proof gives us more information about the greatest common divisor that will be useful later.

Bezout’s original theorem is proved in terms of integers, not natural numbers, using negative numbers and subtraction. We work this out in the terms of natural numbers only. The proof is longer and more complicated, essentially because it replaces subtraction with steps that involve cancellation. The advantages are that the proof demonstrates how to use cancellativity in a proof, and that the result only involves natural numbers (integers are not really needed).

Before moving to the proof of the theorem, we establish a simple lemma.

**Lemma 22:**

For any natural numbers $m$, $n$ and $p$, if $0 < m$, then $p$ is a common divisor of $m$ and $n$ if and only if $p$ is a common divisor of $(n \mod m)$ and $m$.

**Proof:** This follows immediately from the fact that for any $m$, $n$ and $p$, if $p \mid m$ then it is the case that $p \mid n$ if and only if $p \mid (m + n)$.

To prove the main result of this section, we introduce a new definition (only needed in this section) and prove a useful lemma about it.
**Definition 16: Bezout Pairs**

For a pair of natural numbers \((m, n)\), say that a pair of natural numbers \((a, b)\) is a **Bezout pair for** \((m, n)\) if and only if \((b \cdot n) - (a \cdot m)\) is the greatest common divisor of \(m\) and \(n\).

Notice that Bezout pairs depend on order. If \((a, b)\) is a Bezout pair for \((m, n)\), it is almost never the case that \((a, b)\), or even \((b, a)\) is a Bezout pair for \((n, m)\).

**Lemma 23: Condition when Bezout pairs can be swapped**

If \((m, n)\) has a Bezout pair and \(m > 0\), then \((n, m)\) also has a Bezout pair.

**Proof:** Suppose \(m > 0\) and

\[ g = b \cdot n - a \cdot m \]  \hspace{1cm} (13.1)

is the greatest common divisor of \(m\) and \(n\). Notice that \(g\) must be positive, for otherwise \(m = n = 0\). So it is also the case that \(am < b \cdot n\). Hence \(n > 0\) and

\[ g + a \cdot m = b \cdot n. \]  \hspace{1cm} (13.2)

Our goal is to find natural numbers \(a'\) and \(b'\) so that \(g + a' \cdot n = b' \cdot m\).

We consider the two cases: \(a = 0\) and \(a > 0\).

Suppose \(a = 0\). Then the greatest common divisor of \(m\) and \(n\) is a positive multiple of \(n\). But then \(g = n\), \(b = 1\) and \(m\) is a positive multiple of \(n\). In that case, we leave it as an exercise to check that \(a' = (m \div n) - 1\) and \(b' = 1\) constitute a Bezout pair for \((n, m)\). That is, \(g + a' \cdot n = b' \cdot m\).

Suppose \(a > 0\). Then \(a \cdot b \cdot m \leq b\). And because \(b \cdot n > 0\), it is also the case that \(a \cdot b \cdot n \leq a\). Again, we leave it as an exercise to check that \(a' = b \cdot (a \cdot m - 1)\) and \(b' = a \cdot (b \cdot n - 1)\) constitute a Bezout pair for \((n, m)\). □
**Theorem 3**: Bezout’s Theorem

For any natural numbers \( m \) and \( n \) for which \( m \leq n \), there is a Bezout pair for \((m, n)\).

**Proof**: By strong induction on \( n \), we show that for every \( m \), if \( m \leq n \) then there is a Bezout triple for \((m, n)\).

**Strong inductive hypothesis**: Suppose that for some \( k \) it is the case that for all pairs \((i, j)\) so that \( i \leq j < k \), there is a Bezout pair for \((i, j)\).

**Inductive step**: The goal is to find a Bezout pair for any pair \((h, k)\) where \( h \leq k \).

For \( k = 0 \), the only case is \( h = 0 \). And clearly, \((0, 0)\) is a Bezout pair for \((0, 0)\).

Suppose \( k \) is positive and \( h \leq k \). We consider three cases: \( h = 0 \), \( h = k \), and \( 0 < h < k \).

**Case** \( h = 0 \): Evidently \( k \) is the greatest common divisor of \( 0 \) and \( k \). And clearly \( k = 1 \cdot k - 0 \cdot 0 \). So \((0, 1)\) is a Bezout pair for \((0, k)\).

**Case** \( 0 < h \leq k \): Then \( k \mod h < h \). So by the strong inductive hypothesis, there is a Bezout pair \((a', b')\) for \((k \mod h, h)\). That is, \( g = b' \cdot h - a' \cdot (k \mod h) \) is the greatest common divisor of \( k \mod h \) and \( h \), and

\[
g + a' \cdot (k \mod h) = b' \cdot h. \tag{13.3}
\]

By Euclid’s lemma, \( g \) is also the greatest common divisor of \( h \) and \( k \). So it remains to find \( a \) and \( b \) satisfying

\[
g + a \cdot h = b \cdot k.
\]

From Equation 13.3, by adding equals to each side and using the fact that \( k = h \cdot (k \div h) + k \mod h \),

\[
g + a' \cdot (h \cdot (k \div h) + k \mod h) = b' \cdot h + a' \cdot h \cdot (k \div h) \quad \text{(13.4)}
\]

\[
g + a' \cdot k = (b' + a' \cdot (k \div h)) \cdot h. \quad \text{(13.5)}
\]

So \((a', b' + a' \cdot (k \div h))\) is a Bezout pair for \((k, h)\). But \( k \) is positive. So Lemma 23 yeilds the desired pair for \((h, k)\).

\(\square\)
Bezout’s Theorem establishes that the greatest common divisor exists for any two natural numbers \( m \leq n \). But because \( \gcd(m, n) = \gcd(n, m) \), it actually shows that the order does not matter, except with respect to determining Bezout pairs.

An algorithm for calculating Bezout pairs, and therefore \( \gcd(m, n) \), can be extracted from the proof. We can also extract a simpler algorithm for calculating \( \gcd(m, n) \) without finding actual Bezout pairs. This is the basis of the algorithm that Euclid provided.

**Algorithm 16: Euclid’s algorithm**

For natural numbers \( m \) and \( n \), \( \gcd(m, n) \) can be computed via the following:

\[
gcd(0, n) = n \\
gcd(m, n) = gcd(n \mod m, m) \quad \text{for any } m > 0
\]

The least common multiple of \( m \) and \( n \) also exists and relates to \( \gcd(m, n) \) in the following way.

**Theorem 4: Least common multiples exist**

Any two natural numbers \( m \) and \( n \) have a least common multiple. Moreover, if we write \( \text{lcm}(m, n) \) for the least common multiple, then \( m \cdot n = \gcd(m, n) \cdot \text{lcm}(m, n) \).

**Proof:** If \( \gcd(m, n) = 0 \), then \( m = 0 \) and \( n = 0 \). So 0 is their only common multiple, and \( m \cdot n = \gcd(m, n) \cdot \text{lcm}(m, n) \).

Suppose that \( \gcd(m, n) \) is positive and that \( m \leq n \) (or swap \( m \) and \( n \) if not). So \( \gcd(m, n) \cdot p = m \) and \( \gcd(m, n) \cdot q = n \) for some natural numbers \( p \) and \( q \).

Let \( s = \gcd(m, n) \cdot p \cdot q \). Clearly, \( s \) is a multiple of \( m \) and is a multiple of \( n \), and \( \gcd(m, n) \cdot s = m \cdot n \). So \( s \) is a common multiple of \( m \) and \( n \). To prove the result, we must show that \( s \) is the least. That is, if \( r \) is also a common multiple of \( m \) and \( n \), then \( r \) is a multiple of \( s \).

Suppose \( m \cdot p' = r \) and \( n \cdot q' = r \). Then \( r = \gcd(m, n) \cdot p \cdot p' = \gcd(m, n) \cdot q \cdot q' \). So \( p \cdot p' = q \cdot q' \). Let \( t = p \cdot p' \).

By Bezout’s Theorem, find natural numbers \( a \) and \( b \) so that

\[
\gcd(m, n) + a \cdot m = b \cdot n.
\]
So
\[ t \cdot \gcd(m, n) + t \cdot a \cdot m = t \cdot b \cdot n. \quad (13.6) \]

But \( t \cdot a \cdot m = s \cdot a \cdot q' \), and \( t \cdot b \cdot n = s \cdot b \cdot p' \). So Equation 13.6 can be written as
\[ r + s \cdot a \cdot q' = s \cdot b \cdot p' \quad (13.7) \]

Since \( r + s \cdot a \cdot q' \) and \( s \cdot b \cdot p' \) are both multiples of \( s \), \( r \) is also a multiple of \( s \). \( \square \)

We can take the result of this theorem as a definition.

**DEFINITION 17: Least common multiple as an operation**

For any natural numbers \( m \) and \( n \), let \( \text{lcm}(m, n) \) denote the least common multiple of \( m \) and \( n \).

Theorem 4 actually tells us more than mere existence of \( \text{lcm}(m, n) \). In fact, it shows that \( \gcd(m, n) \cdot \text{lcm}(m, n) = m \cdot n \). This is exactly analogous to one of the laws we established in Chapter ??: \( \min(m, n) + \max(m, n) = m + n \). Though the proof of Theorem 4 is much more complicated, the two results are clearly similar. They are in fact closely related. Many of the laws we enumerated for \( \min \) and \( \max \) carry over to greatest common divisor and least common multiple. Compare the following table of laws to the table on page 118.

---

The proof of Theorem 4 can be used to extract an algorithm for computing \( \text{lcm}(m, n) \). The simplest, but not most efficient, method is to take \( \text{lcm}(m, n) = \lfloor m \cdot n \rfloor \div \gcd(m, n) \).
BASIC LAWS OF gcd AND lcm

For any natural numbers m, n and p:

**Characterization via**

\[ m | \text{gcd}(n, p) \text{ if and only if } m | n \text{ and } m | p \]
\[ \text{lcm}(m, n) | p \text{ if and only if } m | p \text{ and } n | p \]

**Associativity**
\[ \text{gcd}(m, \text{gcd}(n, p)) = \text{gcd}(\text{gcd}(m, n), p) \]
\[ \text{lcm}(m, \text{lcm}(n, p)) = \text{lcm}(\text{lcm}(m, n), p) \]

**Commutativity**
\[ \text{gcd}(m, n) = \text{gcd}(n, m) \]
\[ \text{lcm}(m, n) = \text{lcm}(n, m) \]

**Idempotency**
\[ \text{gcd}(m, m) = m \]
\[ \text{lcm}(m, m) = m \]

**Absorptivity**
\[ m = \text{gcd}(m, \text{lcm}(n, m)) \]
\[ m = \text{lcm}(m, \text{gcd}(n, m)) \]

**Distributivity**
\[ m \cdot \text{gcd}(n, p) = \text{gcd}(m \cdot n, m \cdot p) \]
\[ m \cdot \text{lcm}(n, p) = \text{lcm}(m \cdot n, m \cdot p) \]
\[ \text{lcm}(m, \text{gcd}(n, p)) = \text{gcd}(\text{lcm}(m, n), \text{lcm}(m, p)) \]
\[ \text{gcd}(m, \text{lcm}(n, p)) = \text{lcm}(\text{gcd}(m, n), \text{gcd}(m, p)) \]

**Modularity**
\[ m \cdot n = \text{gcd}(m, n) \cdot \text{lcm}(m, n) \]

13.1 Relative Primality

One characterization of prime numbers is that \( p \) is prime if and only if \( p \) is greater than 1 and \( \text{gcd}(p, n) = 1 \) for all \( n \) lying strictly between 1 and \( p \). For example, 5 is prime. It is easy to check that \( \text{gcd}(5, 2) = \text{gcd}(5, 3) = \text{gcd}(5, 4) = 1 \). More generally, pairs of numbers for which \( \text{gcd}(m, n) = 1 \) play an important role in many parts of discrete mathematics. Hence the following definition.
Definition 18: Relatively Prime Numbers

Natural numbers m and n are said to be relatively prime if and only if \( \text{gcd}(m, n) = 1 \).

When you “reduce” a fraction, such as \( \frac{24}{15} \), you find the greatest common divisor \( \text{gcd}(24, 15) \) and factor it from the numerator and denominator. In this case, the result is \( \frac{3 \cdot 8}{3 \cdot 5} = \frac{8}{5} \). The result has a numerator and denominator that are relatively prime. In fact, this is a useful definition of “reduced fraction”. I won’t pursue this idea here because fractions are not our concern for now. Nevertheless, a couple of useful lemmas regarding relative primality are in order. You can work out how these relate to fractions.

Lemma 24: Factoring out a greatest common divisor results in relatively prime numbers.

Suppose m and n are any two natural numbers. Choose p and q so that \( m = \text{gcd}(m, n) \cdot p \) and \( n = \text{gcd}(m, n) \cdot q \). Then p and q are relatively prime.

Proof: Exercise. □

Lemma 25: Products of relative primes are relatively prime

If \( \text{gcd}(m, p) = 1 \) and \( \text{gcd}(n, p) = 1 \), then \( \text{gcd}(m \cdot n, p) = 1 \).

Proof: In case \( p = 0 \), this is trivial since \( \text{gcd}(m, 0) = m \) and \( \text{gcd}(n, 0) = n \). Otherwise, \( \text{gcd}(n, p) = \text{gcd}(p \mod n, n) \) and \( \text{gcd}(m, p) = \text{gcd}(p \mod m, m) \) Suppose \( \text{gcd}(m \cdot n, p) > 1 \), \( \text{gcd}(m, p) = 1 \) and \( \text{gcd}(n, p) = 1 \). So \( \text{gcd}(m \cdot n, p) \) divides p. Evidently, it can not divide m or n. So there is a non-zero remainder in both cases. That is, \( m = q \cdot \text{gcd}(m \cdot n, p) + r \) and \( n = q' \cdot \text{gcd}(m \cdot n, p) + r' \) for some \( q, r, q', r' \) where \( 0 < r, r' < \text{gcd}(m \cdot n, p) \). □
**Lemma 26:** A cancellation law for divisibility

If $m$ and $n$ are relatively prime and $m \mid (n \cdot p)$, then $m \mid p$.

**Proof:** Suppose $\gcd(m, n) = 1$ and $m$ divides $n \cdot p$. Either $m \leq n$ or $n < m$. Suppose $m \leq n$, Bezout’s Theorem yields natural numbers $s$ and $t$ so that $1 + s \cdot m = t \cdot n$. Notice that $t$ can not be 0. So $p + p \cdot s \cdot m = p \cdot t \cdot n$. Since $m$ divides the right side of this equation, it divide the left. Since $m$ also divides $p \cdot s \cdot m$, it must divide $p$. Suppose $n < m$, then again Bezout’s Theorem yields natural numbers $s$ and $t$ so that $1 + s \cdot n = t \cdot m$. In particular, $t \neq 0$. So $m \mid t \cdot n \cdot p$ and $t \cdot n \cdot p = p + s \cdot m \cdot p$. Again $m$ must also divide $p$. □

**Exercises**

99. For each of the following pairs of numbers, calculate their greatest common divisor and indicate which pairs are relatively prime.

100. Recall the Fibonacci numbers. Show that $\text{fib}(n)$ and $\text{fib}(n + 1)$ are always relatively prime.

101. Show that for any natural numbers $m$ and $m \cdot n + 1$ are always relatively prime.

A natural next topic is primality. As you know, every positive natural number is composed of prime factors in essentially one way. For example, 30 is $2 \cdot 3 \cdot 5$, and there is no other decomposition of 30 except by re-ordering the factors. To make this general fact, known as the *Fundamental Theorem of Arithmetic*, precise, it will be helpful to have more mathematical equipment at our finger tips. Also, we now have some experience with proofs. It will help to study how proofs work.
Prime Numbers

Prime numbers are building blocks for the natural numbers. As such they are fundamental to many branches of discrete mathematics.

We all know what a prime number is: a number that is not 1 and has no factors smaller than itself. One might ask why 1 is excluded. After all, 1 does not have any factors smaller than itself either. So at first blush, excluding 1 seems like an arbitrary decision. In fact, historically 1 has sometimes been regarded as a prime number. But that actually makes things more difficult. So now, everyone agrees that 1 should not be considered to be prime. In the next paragraphs, we look at why this makes sense.

Recall that the product of a list of natural numbers is defined inductively by

- $\prod \emptyset = 1$
- $\prod n : L = n \cdot \prod L$

Definition 19: Prime Numbers

A prime number is a positive natural number $p$ so that for any list $\text{LinList}[\mathbb{N}]$ of natural numbers, if $p \mid \prod L$, then $p \mid n$ for some $n \epsilon L$.

With this definition, 1 is not prime for exactly the same reason that 6 is not. In both cases, all we need to show is the number divides some product of natural numbers, but does not divide any of the factors in that product. For 1, just observe that 1 divides $\prod \emptyset$, but 1 does not divide any item on the list $\emptyset$ (because there are no items on the empty list). Likewise, 6 divides $\prod [2, 3]$ but 6 does not divide any item on the
list \([2, 3]\). In contrast, 2 is prime because if \(2 \mid \prod L\) with \(L\) being a list of natural numbers, at least one of the items on the list must be even.

Notice that for both of the counter-examples, 6 and 1, we were able to choose a product that does not just divide the number, but is equal to the number. This suggests that the definition of primality is related to a seemingly more general notion.

**Definition 20: Irreducible numbers**

An **irreducible number** is a positive natural number \(p\) so that for any list \(L\) of natural numbers, if \(p = \prod L\), then \(p \in L\).

This definition is most likely what you were told is the definition of primality. That is, a prime number is usually defined to be any natural number greater than 1 that can not be factored into properly smaller numbers. It is not hard to see that this is precisely what irreducibility means.

The distinction between irreducibility and primality is an important one in other contexts. We will encounter some of those later when we discuss modular arithmetic. As it happens though, in the natural numbers primes and irreducibles are the same. So why bother to have two definitions now?

First, in situations very closely related to the natural numbers, primes and irreducibles really are not the same. So it is helpful to be aware of the distinction now. Second, the proof that the two concepts are the same for natural numbers is instructive. To show that all primes are irreducible is fairly easy. The proof that all irreducibles are prime is more complicated.

**Lemma 27: All primes are irreducible**

Any prime number is also an irreducible number.

**Proof:** Suppose \(p\) is prime. Consider a list \(L\) so that \(p = \prod L\). Then \(p\) divides \(\prod L\). So there is some \(n \in L\) so that \(p\) divides \(n\). But \(n\) also divides \(\prod L\). Because divisibility is anti-symmetric, \(p = n\). \(\Box\)

To prove the converse, we need a fact that (in a different form) was known to Euclid.
LEMMA 28: Euclid’s Lemma

If $p$ is irreducible, then for any $m$ either $m$ and $p$ are relatively prime, or $m$ is a multiple of $p$.

**Proof:** Suppose $p$ is irreducible. In particular, it is positive, so $\gcd(p, m) \neq 0$.

Suppose $1 < \gcd(p, m) < p$. Then for some $q$, $\gcd(p, m) \cdot q = p$.
But then $1 < q < p$ as well. So $p$ is not irreducible. This contradicts the assumption, so $\gcd(p, m)$ must either equal 1 or be greater than or equal to $p$. But $\gcd(p, m) \leq p$ always holds when $p$ is positive.

\[ \square \]

THEOREM 5: Irreducibles are prime

Every irreducible number is prime.

**Proof:** Suppose $p$ is irreducible. By induction on lists, we show that if $p \mid \prod L$ holds for some list $L$, then $p \mid n$ for some $n$ on the list.

**Basis** The basis holds vacuously because $\prod [] = 1 \neq p$.

**Inductive Hypothesis** Suppose $K$ is a list of natural numbers so that if $p \mid \prod K$, then $p \mid n$ for some $n$ on the list $K$.

**Inductive Step** The goal is to show that for any natural number $m$, if $p \mid \prod (m : K)$, then $p \mid n$ for some $n$ on the list $m : K$. That is, either $p \mid m$ or $p \mid n$ for some $n$ on the list $K$.

By Euclid’s Lemma, either $m$ is a multiple of $p$ or $m$ and $p$ are relatively prime. If it is the former, then $p \mid m$. If it is the latter, then $p \mid \prod K$ by Lemma 26. So by the inductive hypothesis, $p \mid n$ for some $n$ on the list $K$.

\[ \square \]
LEMMA 29: All irreducibles are prime

Any irreducible number is a prime number.

PROOF: Suppose \( r \) is an irreducible number. We prove by induction on lists that if \( r \mid \prod L \), then \( r \mid n \) for some \( n \) on the list.

**Basis** If \( r \) is irreducible, it is greater than 1. So \( r \) can not divide \( \prod \). The basis is true vacuously.

**Inductive Hypothesis** Suppose that \( K \) is a list of positive natural numbers so that if \( r \mid \prod K \), then \( p \mid n \) for some \( n \) on the list \( K \).

**Inductive Step** The goal is to show that for any positive \( m \), if \( r \) divides \( \prod (m : K) \), then \( r \) divides some \( n \) on the list \( m : K \). That is, either \( r \mid m \) or \( r \mid n \) for some \( n \) on the list \( K \).

Suppose that \( r \mid \prod (m : K) \). Consider \( g = \gcd (r, m) \). Because both \( r \) and \( m \) are positive, \( g \) is also positive and \( r = g \cdot q \) for some \( q \). Because \( r \) is irreducible, either \( g \geq r \) or \( q \geq r \). So one of the other is equals \( r \). If \( g = r \), then \( r \mid m \). Otherwise, \( r \) and \( m \) are relatively prime. So by Lemma 26, \( m \mid m \cdot \prod K \), \( m \mid \prod K \). So the inductive hypothesis ensures that \( m \mid n \) for some \( n \) on the list \( K \).

\( \square \)

Consequently, we can use the terms “irreducible” and “prime” interchangibly for natural numbers (also for integers when they come up). But bear in mind that the two concepts differ in other contexts.

Our next task is to prove two facts you are been told many times. First, the Fundamental Theorem of Arithmetic says that every positive natural number can be factored in essentially one way into prime numbers. Second, we show that there are infinitely many primes.

We split the proof of Fundamental Theorem into two parts. First, we prove that every positive natural number factors into irreducibles. Then we show that any factorization into primes is unique except for the order in which the primes are listed.
For any positive natural number \( m \), there is a list \( P \) consisting only of irreducible numbers so that \( m = \prod P \).

**Proof:** Here we use strong induction.

- [Strong Inductive Hypothesis] Assume that for some positive \( k \), it is the case that for every \( 0 < j < k \), there is a list of primes \( P \) so that \( j = \prod P \).

- [Strong Inductive Step] There are three cases to consider. Either \( k = 1 \), or \( k = i \cdot j \) for some positive \( i \) and \( j \) strictly less than \( k \), or neither of these holds.

Suppose \( k = 1 \). Then we let \( P = \[] \). This is a (trivial) list of irreducibles whose product is \( k \).

Suppose \( k = i \cdot j \) where \( i \) and \( j \) are both positive and strictly less than \( k \). By the inductive hypothesis, there are lists of primes \( Q \) and \( R \) so that \( i = \prod Q \) and \( j = \prod R \). Hence \( k = \prod Q \cdot \prod R = \prod (Q + R) \).

Suppose \( k \) is neither equal to 1, nor equal to \( i \cdot j \) for any two natural numbers strictly less than \( k \). Then \( k \) itself is irreducible. The proof of this is by induction on lists.

**Basis** Because \( k \neq \prod [i] \), the basis is vacuously true. That is if \( k = \prod [i] \), then \( k \) would appear on the list [i].

**Inductive hypothesis** Suppose that for a list \( K \), if \( k = \prod K \), then \( k \) appears on the list.

**Inductive step** The goal is to show that for any \( n \), if \( k = \prod (n : K) \), then \( k \) appears on the list \( n : K \). That is, either \( k = n \) or \( k \in K \).

By definition, \( \prod (n : K) = n \cdot \prod K \). But the case of \( k \) under consideration is that \( k = i \cdot j \) implies either \( i \geq k \) or \( j \geq k \). Because \( k \) is positive, either \( k = n \) or \( k = \prod K \). In the first case, \( k \in (n : K) \).

In the second case, by the inductive hypothesis for \( K \), \( k \in K \), so \( k \in (n : K) \).

So \( k \) is irreducible. And since \( k = \prod [k] \), it is the product of a list of irreducibles.

These are the only possible cases. So the strong inductive step is proved.

□
The list of irreducibles constructed in Lemma 30 is often called a prime factorization of $m$ (because we know irreducible and prime mean the same thing). Generally, a composite has more than one such factorization for trivial reasons. For example, the lists $[2,3]$ and $[3,2]$ both factor 6 into irreducibles. But the only way two such factorizations of the same number can differ is by the order in which the factors are listed. If we insist that our lists are sorted in increasing order, this ambiguity is avoided.

Recall from Chapter ?? that because $\mathbb{N}$ is ordered by $\leq$, we can say what it means to have a sorted list of natural numbers. Namely, a list $L$ of natural numbers is sorted if $L_i \leq j$ whenever $i \leq j < \text{len}(P)$.

**Lemma 31:**

Sorted lists of primes are determined by their products. For any sorted lists of primes, $P$ and $Q$, if $\prod P = \prod Q$ then $P = Q$.

**Proof:** The proof is by list induction on $P$.

**Basis** For any list $Q$ of irreducibles, if $\prod[] = \prod Q$, then $Q$ must be the empty list as well. I leave the details of checking this to you.

**Inductive Hypothesis** Suppose that for some sorted list $K$ consisting of primes it is the case that for every sorted list $P$ of primes, if $\prod K = \prod P$, then $K = P$.

**Inductive Step** Consider some prime $m$ so that $m : K$ is sorted. The goal is to show that for any sorted list of primes $Q$, if $\prod m : K = \prod Q$, then $m : K = Q$. This is by list induction on $Q$.

**Basis** As before, $\prod[] = 1$. So the basis holds vacuously.

**Inductive Hypothesis** Suppose that for some list sorted list $J$, it is the case that if $\prod m : K = \prod J$, then $m : K = J$.

**Inductive Step** Consider prime $n$ so that $n : J$ is sorted. If $m = n$, then $\prod K = \prod J$. So by the main inductive hypothesis, $K = J$. So $m : K = n : J$.

To complete the inductive step we show that $m < n$ and $n < m$ are both impossible. If $m < n$, then $m$ is relatively prime to $n$. So $m : \prod J$, but because $n : J$ is a sorted list of primes, $m$ is also relatively prime to each item in $J$. This contradicts primality of $m$. Likewise, $n < m$ contradicts primality of $n$ for the same reason by swapping the roles of $m$ and $n$, and of $K$ and $J$.

So we have completed the inductive proof that for any sorted list of
The preceding lemmas show that every positive \( m \) has a unique sorted prime factorization, typically called the prime factorization.

**Theorem 6:** Fundamental Theorem of Arithmetic

Every positive natural number has a unique sorted prime factorization.

**Proof:** All that remains is to the remark that if \( P \) is a prime factorization of \( m \), then \( P \) can be sorted into increasing order. The result has the same product \( P \) because of commutativity. \( \square \)

For example, 24 is factored as \([2, 2, 2, 3]\) and 800 is factored as \([2, 2, 2, 2, 5, 5]\). We can get a more efficient representation by listing the number of times each prime is repeated. So we can represent 800 by \([5, 0, 2]\), signifying that \( 800 = 2^5 \cdot 5^2 \). Notice that in this notation, we need the middle 0 as a “place holder” to indicate that our number does not have any 3 factors. Also “trailing zeros” in this notation do not make a difference. \([5, 0, 3, 0]\) also represents 800. The extra 0 at the end simply tells us that 800 is not divisible by the next prime (7). We will investigate this notation informally (without supplying proofs) after establishing that we have a plentiful supply of primes. The advantage of the notation is that it allows us to see immediately how \( \text{gcd} \) and \( \text{lcm} \) are not merely analogous to \( \min \) and \( \max \), but are actually closely related computationally.

**Theorem 7:**

There are infinitely many primes.

For every natural number \( m \), there is a prime \( p \) greater than \( m \).

**Proof:** We prove this by showing that no finite list of prime exhausts all of them.

Suppose \( L \) is a non-empty list consisting of primes. To show that \( L \) is missing something, consider the number \( m = 1 + \prod L \). Since \( \prod L \geq 1, m > 1 \). So \( m \) has a non-empty irreducible factorization, say \( M \). Clearly \( M \) does not have any item in common with \( L \) (this could be proved explicitly by induction on \( L \)). So any item on the list \( M \) is...
missing from $L$. Since $M$ is not empty, $L$ can not be an exhaustive list of all primes. □

**Definition 21:**

We can enumerate the primes: 2, 3, 5, 7, . . . in increasing order. For every natural number $k$, let $p_k$ be the $k^{th}$ prime. That is, the numbers $p_k$ satisfy

\[
p_0 = 2
\]

\[
p_k (> ) = \text{the smallest prime } q \text{ so that } p_k < q
\]

Notice that this is well-defined because for any $m$ there is a prime greater than $m$. We would not know this if we did not know there are infinitely many primes.

**Exercises**

102. What is $p_{10}$?

103. What is the prime factorization of 1440?

Using $p_k$, we can succinctly represent any positive natural number by a list of exponents of primes. Namely, for a list $R$ of natural numbers, define $R_{pe}$ (for “prime exponent representation”) to be

\[
R_{pe} := \prod_{i < \text{len } R} p_i^{R_i}.
\]

For example, $[1, 2, 0, 2]_{pe} = 2^1 3^2 5^0 7^2 = 882$.

For a positive natural number $n$, let $PE(n)$ denote the unique list so that (i) $PE(n) \neq n$ and (ii) $PE(n)$ does not contain any trailing zeroes.

Define $P \mp Q$ for two lists of natural numbers by adding the items of the two lists itemwise. That is,

\[
P_{+}[] = P
\]

\[
[]_{+} Q = Q
\]

\[
m : P_{+} n : Q = (m + n) : (P_{+} Q)
\]

Then it is clear (we will not give a proof) that $P_{pe} \cdot Q_{pe} = (P_{+} Q)_{pe}$. 
Define a relation $P \preceq Q$ on lists of natural numbers by

- $[\ ] \preceq Q$ always,
- $m : P \preceq [\ ]$ never, and
- $m : P \preceq n : Q$ if and only if $m \leq n$ and $P \preceq Q$.

Then $P_{pe} \mid Q_{pe}$ if and only if $P \preceq Q$. In other words, if we list the exponents of primes that constitute given numbers $m$ and $n$, we can compare them for divisibility simply by comparing the exponents by $\leq$.

This leads to the relation between min and gcd. Namely, define an operation on lists of natural numbers by taking minima itemwise.

$$\overline{\min}(P, [\ ]) = P$$
$$\overline{\min}([\ ], Q) = Q$$
$$\overline{\min}(m : P, n : Q) = \min(m, n) : \overline{\min}(P, Q)$$

Then it is also easy to check that $\overline{\min}(P, Q)_{pe} = \gcd(P_{pe}, Q_{pe})$ for any two lists of natural numbers.

A similar definition of max will lead to $\overline{\max}(P, Q)_{pe} = \lcm(P_{pe}, Q_{pe})$ for any two lists of natural numbers as well.

Now the fact that $m \cdot n = \gcd(m, n) \cdot \lcm(m, n)$ becomes clearly a corollary of the fact that $m + n = \min(m, n) + \max(m, n)$. Namely, for any two lists of natural numbers, $P$ and $Q$,

$$P \mp Q = \overline{\min}(P, Q) \mp \overline{\max}(P, Q)$$

is easily proved by induction on lists. And the preceding paragraphs show that $m \cdot n = \gcd(m, n) \cdot \lcm(m, n)$ follows from this.

In principle, $\gcd(m, n)$ could be calculated by first factoring $m$ and $n$ into primes, then using $\overline{\min}$ on the resulting lists, then converting that back to a natural number. This method, however, is extremely inefficient because finding the prime factorization of a large number is difficult.