Learn You Some Algebras for Glorious Good!

Peter Harpending <peter@harpending.org>

April 26, 2015

Editors & Noted Contributors

Alexander G Bauer <sasha@crofter.org>
Randy Brown <randy@beingbrown.net>
Nick Chambers <DTSCode@gmail.com>
Ng Wei En <wei2912.supp0rt@gmail.com>
Anton Golov <jesyspa@gmail.com>
Ben Z RF <benzrf@benzrf.com>

Please keep in mind that this book is far from finished, and what you are reading is an extremely rough and unfinished draft.

Contents

Chapter 1

Introduction

Before I bore you with a bunch of stuff you don't care about, let's do some math, shall we?

There are essentially three notions with which you need to be familiar in order to do anything interesting in math. These three things are *sets*, *functions*, and *proofs*. Unfortunately, to be familiar with one, you have to be familiar with the other two.[1]

So, what are each of those things?

- A *set* is an unordered collection of things. There is also no repetition. For instance, $\{2,5\}$ is the same as $\{5,2\}$ (because order doesn't matter). $\{2,5,5\}$ would be the same set, because there's no notion of multiplicity.

  You see what I did there? I subtly introduced some notation: the bracket notation for sets. This math stuff really isn't all that difficult. It's also really interesting if you approach it the right way.

- A *function* is a mathematical construct (well, obviously, else I wouldn't be talking about it). Basically, it takes some input, does something to it, and spits out some output. If you give the function the same input a bunch of

[1] You'll learn as we go along, when math people use a common term like *set*, *function*, *proof*, *group*, *continuous* or *closed*, they usually mean something similar in concept to the colloquial term, but there are some strings attached. This is usually the case in the sciences too (e.g. *theory*, *hypothesis*, *experiment*).

times, you should get the same result each time. This concept is called "referential transparency." If the function is not referentially transparent, then it's not a function. It's something else.

- A *proof* is basically where you take a bunch of simple facts, called *axioms*, and chain them together to make *theorem*s. It's sort of like sticking puzzle pieces together to form a picture.

  The puzzle pieces (in this case, the axioms) aren't usually very interesting on their own. However, the picture they form (in this case, the theorem) can be really cool and enlightening. The proof would be analogous to an explicit set of instructions explaining how to put the pieces together.

Once you are familiar with each of those concepts, we can do all sorts of cool stuff. Throughout the book, we will prove all of the following:

- If you tap your finger against a bridge at exactly the right frequency, the bridge will collapse. (Resonance)
- The formula used to calculate the interest rate on your mortgage is actually just a fancy form of the ratios of angles in a triangle. (Euler's formula)

1.1 How to read the book

The best way to read this book is to just read it. Don't skip sections, or look ahead, or anything like that. Just read it straight through. It's also pretty important that you read the rest of this chapter. I promise it's not too boring.

Do all of the exercises. There aren't that many. However, they are pretty difficult. The exercises all have solutions, which are in § D.

The exercises are designed to make you think, and widen your perspective on the topic at hand. They are not designed to be tedious. They are difficult, but the good kind of difficult.

It would be perfectly okay to just do the exercises (all of them), and then go back and read the text when you don't understand something.

§ A is a reference section. It contains every single theorem, definition, identity, and property in this book. Unlike the contents of this book, § A is not meant to be read straight through. However, if you don't remember the name of something, or want to know if some property is true, § A is the place to look.

My writing strategy involves writing the bare minimum information, with criminally few examples or exercises, so I can get the structure right, then to go back and fill in the blanks. So, until this book is finished, it's going to be horrifyingly fast paced.

1.2 Introduction (for real this time)

This is a math book. Well, duh. Why did I write it?

Most math (and science) books nowadays seem to value keeping an academic tone over ensuring that the reader understands the material, and — more importantly — enjoys reading the book.

I take the opposite approach. I want to create a book that is fun to read and easy to understand, while eschewing the practice of making myself look good.

In high school, and even in college, I noticed that subjects were taught with pragmatism in mind: here's what you need to learn to have a job doing X. It completely zaps the intellectual curiosity of the students. I think that's a shame. I hope reading this book will rekindle your intellectual curiosity.

The inspiration for this book is *Learn You a Haskell for Great Good!*, by Miran Lipovača. Haskell is a programming language, and LYAH is a great book for learning Haskell. If you are interested in a print copy of LYAH, see [15].

There is also an incomplete and unofficial Russian translation (`https://github.com/gazay/lysa`), courtesy of Alexey Gaziev.

1.2.1 Programming

In this book, I cover a lot of hard stuff.[2] Sometimes, it's useful to program your way through a problem. You don't have to, but it helps. Every programmer will tell you that programming teaches a manner of thinking.

Many programmers will cite Steve Jobs's[3] famous quote, regarding the use of programming in his job,

> *[sic] . . . much more importantly, it had nothing to do with using [the programs we wrote] for anything practical. It had to do with using them to be a mirror of your thought process; to actually learn how to think. I think everybody in this country should learn how to program a computer — should learn a computer language — because it teaches you how to think.*

That first sentence or two is actually a pretty good description of mathematics (and programming). Both are incredibly useful, and have endless practical applications.

That's not the point, though. The whole usefulness thing is a side gig. It's about learning how to think, and having a rigorous language through which to express your thoughts.

Furthermore, the rigor of the language helps you build upon your current thoughts to find out even cooler things. That's what math (and programming) is about.

Sage (`http://www.sagemath.org/`) is a mathematically-oriented programming language. If you're familiar with Mathematica or Maple, Sage is an open-source alternative.

Sage is perhaps most useful in generating graphs transparently. Sage is also capable of solving your equations, finding holes in your logic, the whole 9 yards.

I would include instructions for installing Sage. However, it takes up a lot

[2] This isn't actually true. Math isn't hard, stupid!

[3] For you youngsters, Steve Jobs is the former CEO of Apple. He's dead now.

of space, and my instructions would quickly be obselete. You can find better instructions for installing Sage on their website (`http://wiki.sagemath.org/DownloadAndInstallationGuide`).

1.3 The community

Despite the fact that I used "I" in the first part of the book, LYSA is actually a community project, and many people participate in the writing of this book.

If you want to talk to us, or to other math people, come see us in `#lysa` on Freenode. If you don't know what IRC is, or you don't have a client set up, you can connect through Freenode's webchat (`http://webchat.freenode.net/?channels=lysa`).

If you have any questions about LYSA (or math), feel free to ask in the IRC channel (`#lysa` on FreeNode in case you forgot).

If you want to submit a correction, or have some issue, or want to add some content, really anything having to do with the content of the book, you can visit our GitHub page (`https://github.com/learnyou/lysa`). We also have a website (`http://learnyou.org`) and a community on Reddit (`https://lysa.reddit.com/`).

1.3.1 Freedom

This book is open-source. That means a number of things:

1. You are free to use this book as you like.
2. You are free to modify the book as you wish. The source is available at `https://github.com/learnyou/lysa`.
3. You are free to redistribute and/or sell exact copies of this book as you wish.
4. You are free to redistribute and/or sell modified copies of this book as you wish.

If, for instance, you are a schoolteacher and want to use this for your class, you are free to edit it to your liking and give the modified copy to your students. The only string I attach is, you have to allow anyone to whom you give the book do the same thing (i.e. they have to be free to copy/modify/change your version).

1.4 Target audience

The explanation of why programming is useful is a good segue into discussing the target audience.

When I was first writing the book, I wrote it in an effort to strengthen my own understanding. So, the target audience was me. The very first versions of this book were about a abstractish branch of math called commutative algebra. Later on, it seemed more fitting to abstractly go over the basics of math. That's what the current version of the book does.

That doesn't answer the question: who is the target audience? Well, people who want to learn basic and intermediate algebra, and to learn why it's so interesting.

I mentioned this in § 1.2. Most books (and people) treat math as a tool you can use for calculations. I treat math as a language you can use to express your ideas. That's the core difference. This book will hopefully give you an interest in math itself, rather than just a cursory knowledge of it.

With that in mind, my book is going to approach the topics much differently than other books on the same topic. I rely very much on abstraction and intuition.

It's that you know all of the basics about arithmetic: you should know how to add, subtract, multiply, divide, and exponentiate. You should also be familiar with the following sets, although it's not really hard to pick up:

1. $\mathbb{N} := \{0, 1, 2, 3, 4, \dots\}$
2. $\mathbb{Z} := \{\dots, `4, `3, `2, `1, 0, 1, 2, 3, 4, \dots\}$
3. The real numbers $\mathbb{R}$.

4. The rational numbers $\mathbb{Q}$.
5. The irrational numbers $\mathbb{I} := \mathbb{R} \setminus \mathbb{Q}$.
6. The complex numbers $\mathbb{C}$.

If not, you should look at § C. § C is more or less a blatant ripoff of [14]. I would like to think that my appendix is much less dry, and does a bit more in the explaining department. Landau's wonderful book is very dry. It's just theorem after theorem after theorem. It's very rigorous, but you'll fall asleep after reading a page.

1.5 Conventions used throughout

You don't actually have to read this section, but it would be useful.

1. `Things in monospace` are either code snippets or commands to be run in a terminal. I have separate stylings for `terminal commands` and `inline code snippets`. That said, they are separate but equal, at least for the time being.
2. The § symbol refers to a section. So § 3.2 means "chapter 3, section 2".
3. Even though most of the writers are American, I still use the British convention of putting periods after quotation marks: "like this". The British convention is less ambiguous. If you see the American convention anywhere in this book, please report it.
4. I will often recommend software. However, I will not recommend any non-libre software, or any software that costs money.
5. "I" refers to me. "We" refers to both me and you, the reader. "You" refers to, you guessed it, the reader. It's the convention in academia to use the so-called "royal we", such as "we subtract 2 from both sides of the equation to obtain the result . . . ".

   Sometimes, we will accidentally use the royal we, out of habit. Crap, I just did it there! See? It's very difficult to avoid. Like any of the other conventions herein, if you see it broken, please report the error to the authors. You

can use the bug tracker (`https://github.com/learnyou/lysa/issues/new`), or, if you don't want to make a gratis GitHub account, you can email me at `peter@harpending.org`.

6. Oh yeah, sometimes I'll use `monospace` in things like URLs or emails for the sake of disambiguity.

7. Most of the authors use some version of Linux. Hence, when there are instructions for computer things (such as installing Haskell), I'll write instructions for Linux, because that's what I know. There are two solutions here:

   (a) You could try out Linux (it's gratis, and it's easy).

   (b) If you know how to do the thing on your OS, and there aren't instructions for your OS, you could write up instructions and add them to the book. If you don't know how to do that, you could bring it up in the bug tracker (`https://github.com/learnyou/lysa/issues/new`) or email me at `peter@harpending.org`.

8. If you see some number as a superscript in the middle of text: like this[4], then the number refers to a footnote. If the superscript number is in the middle of math, it's probably just math.

9. If there's some number in brackets, like this: [15], then it's a citation. If you're reading this as a PDF, you can actually click on the number, and your PDF reader will take you to the relevant bibliography entry. Go ahead, check it out! I'll wait. You can do the same thing for footnotes and URLs.[5]

10. If you see something *in italics*, it's usually a vocabulary word. Often there will be a term with a section number next to it: *like this* (§ A.1), usually somewhere in § A. § A is a reference section, which has theorems, vocabulary, identities, stuff like that. So, the reference to a section in § A next to a term points to the relevant section in the appendix. Like all of the other references — citations, footnotes, URLs — you can click on the section title, and your PDF reader will take you there.

[4] Hey, you found me!

[5] Well, clicking the URL will open up your web browser, but you get the point

Chapter 2

Booleans, propositional logic, and simple operators

Before we get into interesting content, you have to understand some stuff. This stuff is pretty easy. This will likely be one of the shorter and easier chapters in the book.

Most people think math is about dealing with numbers and pumping out formulas. Well, that's not what math is about. As said in § 1, it's about using math as a language to express your thoughts. Most people don't think about numbers all day; thus, we deal with things in math that aren't numbers.

In this next section, we're going to outline some basic rules for reasoning about things. You need to know these rules to do really cool stuff. Although, as you will (hopefully) see, these rules can be fun to toy around with on their own.

It's okay if you don't remember all of these rules. You can always find a list in § A.3.

2.1 Propositions

The first thing you need to understand is the notion that "if x is true, then y is also true. But, if y is true, it's not necessarily true that x is false." As always,

mathematicians are too lazy to write this stuff out by hand, so they have notation for it.

1. $a \implies b$ means that "a implies b". It doesn't necessarily mean that b implies a. It means that if a is true, then b is also true.

   Here's an example:

   If someone is decapitated, then they will die. So,

   $$\text{Decapitated} \implies \text{Dead}$$

   However, if someone is dead, it doesn't necessarily mean that they were decapitated. They could have been shot, or stabbed, or had a heart attack. There are endless possibilities.

2. $a \impliedby b$ is the same as writing $b \implies a$. It's sometimes convenient to use $a \impliedby b$ instead. $a \impliedby b$ should be read "a is implied by b".

3. When I strike through some mathematical operator, like this: $\not\Rightarrow$, it means that you can semantically put "not" in front of whatever the operator says. So, $\not\Rightarrow$ means "not implies". That doesn't make much grammatical sense in English, so "does not imply" might be better. Nonetheless, you get the point.

   Moving on from the example above:

   $$\text{Decapitated} \implies \text{Dead}$$

   If someone is decapitated, then they're also dead (at least within a few seconds). However, if someone is dead, it's not necessarily true that they were decapitated.

   $$\text{Decapitated} \not\impliedby \text{Dead}$$

If it were the case that Decapitated $\not\Longrightarrow$ Dead

4. If something is not true, then I'll put a $\neg$ in front of it. So, if I want to say that a is false, then I'll write $\neg a$.

5. If I want to pose a question, I could just ask the question. For instance, "Is $\neg$Decapitated $\Longleftarrow$ $\neg$Dead true?".

   However, that quickly becomes difficult, usually when there are multiple assertions in a mathematical expression, and you don't know which one I'm asking about. Moreover, since I use the same font for text and math, if I have both, it might be hard to tell which is math and which is text. So, to help with ambiguity, I'll put a ? over the operator I'm asking about:

   $$\neg\text{Decapitated} \overset{?}{\Longleftarrow} \neg\text{Dead}$$

   See, that's much easier.

6. Now, on to that question - Is "not decapitated" implied by "not dead". Well, let's think about it. If someone is not dead, then they couldn't have been decapitated, because if they were decapitated, then they would be dead. Therefore, if someone is not dead, then they weren't decapitated.

   That word jumble was probably confusing. Mathematicians don't like to be confused. I'll make it symbolic for you.

If

$$\text{Decapitated} \implies \text{Dead}$$

then

$$\neg\text{Decapitated} \impliedby \neg\text{Dead}$$

This is another rule about propositional logic:

If

$$a \implies b$$

then

$$\neg b \implies \neg a$$

2.2 Booleans as a monoid

Okay, first of all, what the hell does the section title mean?

Let's go over it:

1. *Booleans* the values true and false. "Boolean algebra" is what we are doing in this section. Dealing with Booleans.

   Why are you capitalizing the word Boolean?

   Because the term is named after a guy named George Boole, who studied them to extent.

2. A *monoid* is a set of things, where you can combine the individual things together. Additionally, there has to be a *monoidal identity*.

   That doesn't help me.

   Well, let me explain it at the end of the chapter.

So, sometimes we need to combine two pieces of logic together. There are two ways we can do this - logical-or and logical-and.

I put logical- in front of them, because the mathematical meaning is slightly different than the colloquial meaning.

Mathematicians are lazy, so we don't like to write "logical-and" whenever we want to say it, so instead we use the symbol $\wedge$.

$A \wedge B$ is true if (and only if) both A and B are true. If one of them is false, then the entire thing is false.

On the other side, we have logical-or. The symbol for logical-or is $\vee$. $A \vee B$ is true if either A or B is true, or if both of them are true. You could think of logical-or as being equivalent to the colloquial "and/or".

If you're having trouble remembering which symbol is logical-and and which one is logical-or, remember that the logical-and symbol — $\wedge$ — looks vaguely like an A.

More formally

There are formal ways to define $\neg$, $\wedge$ and $\vee$

I'm going to introduce a symbol: $:=$. It means "is defined to be equal to".

$$x := y$$

means "x is defined to be equal to y".

With that in mind, here is a more formal definition of $\wedge$:

$$\begin{array}{rcrcl} \text{False} & \wedge & \_ & := & \text{False} \\ \text{True} & \wedge & x & := & x \end{array}$$

The _ just means "you can ignore whatever goes here".

Likewise, for $\vee$:

$$\begin{array}{rcrcl} \text{True} & \vee & \_ & := & \text{True} \\ \text{False} & \vee & x & := & x \end{array}$$

and finally for $\neg$:

$$\begin{array}{rcrcl} \neg & & (\neg x) & := & x \\ \neg & & \text{True} & := & \text{False} \end{array}$$

2.2.1 Properties of the Boolean operators

There are a lot of properties about $\neg$, $\wedge$, and $\vee$. Most of them are pretty intuitive, so you won't have trouble memorizing them. You should try to remember the names of the properties, but it's not really that important.

Cancellative property For all Boolean values a,

$$\neg\neg a = a$$

I mentioned this before somewhere in § 2.1. This is so simple that it's impossible to prove. Thus, it's called an *axiom* or *postulate*.

Reflexive property For all Boolean values a,

$$a \wedge a = a$$

Likewise, for all a,

$$a \vee a = a$$

This is called the *reflexive property*.

Proof. □

Associative property For all a, b, and c,

$$a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c$$

Commutative property For all a and b:

$$a \wedge b \equiv b \wedge a$$

Distributive property $a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$

De Morgan's first law $\neg (a \wedge b) \equiv \neg a \vee \neg b$

Reflexive property $a \vee a \equiv a$

Associative property $a \vee (b \vee c) \equiv (a \vee b) \vee c$

Commutative property $a \vee b \equiv b \vee a$

These values, true and false, are called *Booleans*. They are named after a mathematician named George Boole who studied them to extent.[4]

2.2.2 If (and only if)

I'm going to introduce some new notation: the $\iff$ symbol.

$$(A \iff B) \iff (A \implies B) \wedge (A \impliedby B)$$

Okay, what the hell is that?

If

$$A \implies B$$

logical-and

$$B \implies A$$

then

$$A \iff B$$

Likewise, if

$$A \iff B$$

then

$$A \implies B$$

logical-and

$$B \implies A$$

The symbol $\iff$ should be read as "if (and only if)". Sometimes I'll write the word iff — with two 'f's — that's the same as "if (and only if)".

In order to do the exercises (yes, there are exercises), you need to know these properties of $\implies$.

1. For all a

   $$a \implies a$$

   So, if a is true, that means that a is true. Duh.

2. For all a, b, and c,

   If

   $$a \implies b$$

   and

   $$b \implies c$$

then

$$a \implies c$$

For this reason, we can write $a \implies b \implies c$ without any ambiguity.

Sometimes, mathematicians, such as myself, will use the symbol $\forall$ in the place of "for all".

2.2.3 Exercises

Ex. 1 — $\neg(A \vee B) \overset{?}{\equiv} \neg A \wedge \neg B$
I'll spoil it for you. This is called *De Morgan's second law*, and it's true. You can prove it using the cancellative property and the first law.

Ex. 2 — Here's another one for you.

$$a \vee (b \wedge c) \overset{?}{\equiv} (a \vee b) \wedge (b \vee c)$$

You need the distributive property, as well as the proof from ex. 1.
While we are at it, these proofs can be found in § A.3, as well as the solutions to these problems.

Ex. 3 — $\neg A \implies B \implies \neg C$

$$C \overset{?}{\implies} A$$

Ex. 4 — $A \not\implies B \implies \neg C$

$$A \overset{?}{\implies} C$$

Ex. 5 — $A \implies B \impliedby C$

$$A \overset{?}{\iff} C$$

Ex. 6 — $A \wedge \neg (B \wedge (C \vee D)) \stackrel{?}{\equiv} A \wedge (B \vee C) \wedge (B \vee D)$

2.2.4 Wait, are you going to tell me what a monoid is?

Okay, you got me!

$\{\text{True}, \text{False}\}$, along with the operator $\wedge$ is a monoid. Additionally, $\{\text{True}, \text{False}\}$, along with the operator $\vee$ forms a monoid.

The "monoidal identity" just means that if you take any a, and $\wedge$ it with True, you're going to get a back again.

The same thing happens with $\vee$ and False

For all Boolean values a,

$$a \wedge \text{True} = a$$

Likewise, for all Boolean values a,

$$a \vee \text{False} = a$$

Chapter 3

Sets

In math, it's often useful to consider *collections* of objects. There are basically two types of collections: *sets* and *vectors*. Sets are unordered, and multiplicity doesn't matter. Vectors are ordered, and multiplicity does matter.

For instance, $\{3,4\}$ and $\{4,3\}$ are the same *set*, but $(3,4)$ and $(4,3)$ are different *vectors*.

Likewise, $\{20,38\}$ and $\{38,20,38,20,20,20,20,20\}$ are the same *set*. You guessed it, $(20,38)$ and $(38,20,38,20,20,20,20,20)$ are different *vectors*. Sets are — at least ostensibly — much more important. More importantly, they are much easier to understand.

Sets were first studied to extent by Georg Cantor, a German mathematician, in the second half of the nineteenth century. Back in his own day, the results Cantor found by studying sets were considered so thoroughly bizarre that many of his colleagues simply refused to believe that Cantor could be right. In the end, Cantor turned out to be right all along. His ideas can be found in any introductory text on mathematics—including this one.

You probably figured it out from above: the notation is $\{$Braces$\}$ for sets, and $($Parentheses$)$ for vectors.

If you can't remember whether to use braces {the curly things}, or parentheses (the round things), remember: *a **brace** is used to **set** a broken bone.* I don't have a horrible pun having to do with parentheses and vectors, and for that, I apologize.

3.1 Elements

Let's invent a set.

$$Q = \{7, 7, 9, 5, 10, 1, 6, 6, 2, 10\}$$

There we go. Remember, order and multiplicity don't matter. But, for the sake of clarity, let's put the elements in order, and deduplicate them.

$$Q = \{1, 2, 5, 6, 7, 9, 10\}$$

Yay! You may have noticed that I slipped in the word *element* into the previous sentence. Objects in the set are called *elements*. Yay, we figured out what that word means!

It's too strenuous on our weak mathematical hands to write "10 is an element of Q", so instead we have the symbol $\in$. $\in$ is a very terrible attempt at drawing an E. If you can't remember what $\in$ is, think "$\in$lement of".[1]

So, I'm going to ask you a question:

$$11 \stackrel{?}{\in} Q$$

(See, I used that thing from earlier with the question mark. I told you it would help.) Well, the answer is no, 11 is not an element of Q. As always, mathematicians are too weak to write "11 is not an element of Q" every time they want to say it, so instead they write

$$11 \notin Q$$

By contrast,

[1] You better think this, because it took me 30 minutes to get the alignment right. So, you know, remember $\in$ this way, or else...

$$6 \in Q$$

What if we want to say "both 6 and 2 are elements of Q"? Well, again, we could write it out like:

$$(6 \in Q) \wedge (2 \in Q)$$

But that's too cumbersome, so instead we'll write

$$2,6 \in Q$$

But won't that get confusing? Only if we put parentheses or braces around 2 or 6.

$$\{2,6\} \in Q$$

That's confusing, don't do that (yet).

There's one more thing I need to go over, which is the null set - it's the simplest set, as it contains no elements. "Null set" takes too long to write, so we use $\varnothing$ instead.

3.2 Subsets and Supersets

Remember when I said $\{2,6\} \in Q$, and we were really confused? In case you don't remember, $Q = \{1,2,5,6,7,9,10\}$. $\{2,6\}$ is obviously not an element of Q. However, $\{2,6\}$ is *in* Q but it's not an element. It's weird. How do we express this notion?

The answer is with *subsets*. "sub" means "smaller", so a "subset" would be a "smaller set". A is a subset of B is all of the elements in A are also in B. The notation is $A \subseteq B$. Some people will read that as "A is contained in B".

Referring to the previous example,

$$\{2,6\} \subseteq Q$$

Wait, what is with the little line under the round half circley thing? So, actually, there are two types of subsets - *proper* and *improper*. $A \subseteq B$ is for improper subsets. $A \subset B$ is for proper subsets. What's the difference, then?

$A \subseteq B$ allows for the possibility that $A = B$. $A \subset B$ means that B is *strictly larger* than A; there are some elements in B that are not in A.

I've already defined $\forall$ in **??**. I'm now going to add another symbol, $\exists$, which means "exists". I mention $\forall$, because $\exists$ is used in the same context.

Anyway, back to business. I use $\subseteq$ for improper subsets, and $\subset$ for proper subsets. However, some people will use $\subset$ for improper subsets, and something like $\subsetneq$ or $\subsetneqq$ for proper subsets. You have to look out.

Here's something cool: $\varnothing$ has no elements, so it's a subset of every set.

$$\varnothing \subseteq A;\ \forall A$$

I'm sure you can figure this out, two sets are equal iff they have the same elements. $A \subseteq B$ means that B has all of the elements that A has. $A \supseteq B$ would mean that A has all of the elements of B. So if both of those are true, then $A = B$. That is:

$$A = B \iff A \subseteq B \wedge A \supseteq B$$

So, what did we learn?

1. There are unordered collections with no multiplicity, called *sets*.
2. There are ordered collections with multiplicity, called *vectors*.
3. Given an object o and a set A, you can ask $o \stackrel{?}{\in} A$.

4. You can pull smaller sets out of a set (I'll show you the mechanics below). The way to express that a set is "embedded" in another set, without being an element is with *subsets*. $A \stackrel{?}{\subseteq} B$ would be asking "are all of the elements in A also in B?". (The converse doesn't necessarily have to be true).

5. There's a pretty easy set which has no elements, called $\varnothing$. An interesting property of $\varnothing$ is that it is a subset of all other sets.

6. Two sets are equal iff they have the same elements.

Chapter 4

Functions

As promised, this chapter discusses functions.

So, what is a function?

So far, we've been dealing with *values* - like 2, $\{3,2,5\}$, and 90. They are static. Static things are fine, but they aren't very interesting. It's much more interesting to examine *changing things* — more specifically, things that change *predictably* and *transparently*.

Enter the *function* (§ A.5). It's a mathematical construct. A function takes some input, and maps it to an output. Functions are sometimes referred to as *mappings* or *morphisms*.

Let's look at a simple function, which takes a number and adds 2 to it

$$\begin{aligned} &f : \mathbb{Z} \to \mathbb{Z} \\ &f = \lambda\,(x) \to x+2 \end{aligned}$$

Pretty simple, right? Okay, so what happens when we send 28 to f?

$$\begin{aligned} f\,(x=28) &= \lambda\,(x=28\,) \to 28+2 \\ &= 30 \end{aligned}$$

Alternatively, since it's obvious we're working with x:

$$\begin{aligned} f(28) &= 28+2 \\ &= 30 \end{aligned}$$

This highlights an important property of functions: *referential transparency* (§ A.5). If you send a function the same input twice, you should get the same output both times. That is,

$$a = b \implies f(a) = f(b)$$

Note that the opposite is not always true:

$$a = b \not\impliedby f(a) = f(b)$$

(If that is true, then the function is *injective*).

Using the lambda — λ — is common when I am using a function without giving it a name. However, usually I will use this notation:

$$\begin{aligned} &f : \mathbb{Z} \to \mathbb{Z} \\ &f(x) = x+2 \end{aligned}$$

The whole $f : \mathbb{Z} \to \mathbb{Z}$ thing should be pretty obvious. If not, it means that f is a function that takes a member of $\mathbb{Z}$ (the whole numbers, both negative and positive), and takes it to another member of $\mathbb{Z}$. Other people might use the notation

$$\mathbb{Z} \xrightarrow{f} \mathbb{Z}$$

That notation is undoubtedly easier to understand. However, as we'll see, that notation quickly becomes unfeasible.

With this in mind, if $f : A \to B$, then A is the *domain* of f, written $\mathbf{dom}(f)$, and B is the *codomain* of f, written $\mathbf{codom}(f)$.

With regard to the function we were just discussing

$$f : \mathbb{Z} \to \mathbb{Z}$$
$$f(x) = x + 2$$

$\mathbb{Z}$ is both the domain and the codomain. If this is the case, then we say that f is a *closure*.[1] f is "closed under $\mathbb{Z}$", meaning that things can't use f to escape from $\mathbb{Z}$. f is closed.

If you can't remember all of these terms, don't worry, they are all listed in § A.5.

4.0.1 Functions with multiple arguments

Remember my explanation of vectors earlier? If not, vectors are like sets, but order and repetition matter.

Here's a function that takes two arguments, and adds them to each other.

$$f : (\mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$$
$$f(x, y) = x + y$$

Pretty easy to understand, right?

If you haven't figured it out from the context, the inputs to the function are called the *arguments*.

Here's a similar function that takes three arguments and adds them to each other

$$f : (\mathbb{Z}, \mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$$
$$f(x, y, z) = x + y + z$$

[1] There's a programming language called Clojure, whose name is a pun on this concept.

You can name your function anything you want, same with the arguments (it doesn't have to be f). It's just a common convention, which you don't have to follow.

What if I want to add a bunch of things together?

Good idea!

$$f : (\mathbb{Z}, \mathbb{Z}, \ldots, \mathbb{Z}) \to \mathbb{Z}$$
$$f(x_1, x_2, x_3, \ldots, x_n) = x_1 + x_2 + x_3 + \cdots + x_n$$

That however isn't ideal, because we have no guarantee that the arguments in the ... are actually integers. How about we have a *set* of integers, and we just take the sum? This has the added benefit of less typing

$$f : \mathbf{Set}(\mathbb{Z}) \to \mathbb{Z}$$
$$f(s) = \sum s$$

So,

$$\begin{aligned} f(\{1,2,3,4,5\}) &= \sum \{1,2,3,4,5\} \\ &= 1+2+3+4+5 \\ &= 15 \end{aligned}$$

4.0.2 Eta-reductions

Mathematicians like to make themselves look smart. One such way is to invent fancy terms for simple things. One such term is the η-reduction.

Let's look at that function we just had

$$f : \mathbf{Set}(\mathbb{Z}) \to \mathbb{Z}$$
$$f(s) = \sum s$$

Notice that we are repeating s on both sides of the equation. It would seem much simpler, and just as clear, to write:

$$f : \mathbf{Set}\,(\mathbb{Z}) \to \mathbb{Z}$$
$$f = \sum$$

That's all an η-reduction is: if you see an extraneous argument, you remove it to make things simpler. As long as we have the signature — the $f : \mathbf{Set}\,(\mathbb{Z}) \to \mathbb{Z}$ thing – it's pretty clear what f does. This is a prime example of mathematicians being both lazy and pretentious at the same time: a practice designed to allow us to be lazier, to which mathematicians have assigned a ridiculous name to make it sound hard.

What the hell is η?

η is the Greek letter eta; it's pronounced "eight-uh".

The ancient Greeks were too dumb to comprehend the concept of "eight". Every time someone brought it up, they said "uh" immediately thereafter. The sound "eight-uh" became so common that they decided to make it a letter.

The Greeks' poor comprehension of simple mathematics remains to this day, and is largely the reason for their current financial crisis.[10]

If you ever take a physics course, you will undoubtedly notice that Greek letters are used frequently in physics. This is the physicists way of subtly hinting that they actually have no idea what they are talking about, and pleading for help from the mathematicians.

4.0.3 Other lambda calculi

This entire idea where you take simple concepts and make them sound really fancy is called λ *calculus* (§ A.6). If you hear people talk about "calculus", they are talking about something else, not this. Nobody is pretentious enough to actually talk about λ calculus.

Anyway, here's a brief summary of λ calculus. You can find this in § A.6, too.

You might want to brush up on your Greek alphabet. I have a nice table of Greek letters in § A.7.

λ **abstraction** A way to write a function: $\lambda\,(x,y) \to x+y$

α **conversion** Changing the names of the arguments. For instance, you can write the above function as

$$\lambda\,(a,b) \to a+b$$

β **reduction** Partially calculating a result. For instance

$$\lambda\,(2,y) \to 2+y$$

Can be β reduced to

$$\lambda\,(y) \to 2+y$$

η **conversion** Removing or adding extraneous free arguments. The last function

$$\lambda\,(2,y) \to 2+y$$

Can be η *reduced* to

$$2+$$

Which could then be η *abstracted* to

$$\lambda\,(2,\kappa) \to 2+\kappa$$

4.1 Currying

We sort of got side-tracked by toying around with sets and making fun of physicists. Hopefully that introduction introduced you to the basic concept of a function, and let you know that they can take multiple arguments

Let's look at that function again:

$$f : (\mathbb{Z}, \mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$$
$$f(x, y, z) = x + y + z$$

What if you wanted to bind $x = 3$, but leave the rest "free"?

$$f : (\mathbb{Z}, \mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$$
$$f(x = 3, y, z) = 3 + y + z$$

Okay, cool. We now have another function:

$$f(3) : (\mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$$
$$f(3, y, z) = 3 + y + z$$

So, actually, instead of needing 3 integers to do its job, f only needed one. However, instead of spitting out another integer, it spit out a function. So, we could write f's signature as:

$$f : \mathbb{Z} \to ((\mathbb{Z}, \mathbb{Z}) \to \mathbb{Z})$$
$$f(x, y, z) = x + y + z$$

Okay, that's sort of weird and unintuitive. Let's try writing f differently:

$$f : \mathbb{Z} \to ((\mathbb{Z}, \mathbb{Z}) \to \mathbb{Z})$$
$$f = \lambda(x) \to (\lambda(y, z) \to x + y + z)$$

Let's look at the second half of that:

$$\lambda(y, z) \to x + y + z : (\mathbb{Z}, \mathbb{Z}) \to \mathbb{Z}$$

(This assumes that we know what x is)

Let's try splitting this up again:

$$\lambda\,(y) \to (\lambda\,(z) \to x+y+z) : \mathbb{Z} \to (\mathbb{Z} \to \mathbb{Z})$$

You give this function a value for y, and instead of giving you a value, it gives you another function, hence the signature $\mathbb{Z} \to (\mathbb{Z} \to \mathbb{Z})$.

Let's plug this back into f:

$$\begin{array}{l} f : \mathbb{Z} \to (\mathbb{Z} \to (\mathbb{Z} \to \mathbb{Z})) \\ f = \lambda\,(x) \to (\lambda\,(y) \to (\lambda\,(z) \to x+y+z)) \end{array}$$

So, instead of f taking three integers, it now only takes one, but spits out a function, which in turn spits out a function, which spits out an integer.

This idea of making a function into a chain of functions is called "Currying".[5] It's named after a dead mathematician named Haskell Curry (ca. 1900-1982), who developed the technique. The programming language Haskell is also named after Mr. Curry.

Getting back to that function, those parentheses are somewhat burdensome, let's get rid of them

$$\begin{array}{rcl} f & : & \mathbb{Z} \to \mathbb{Z} \to \mathbb{Z} \to \mathbb{Z} \\ f & = & \lambda\,(x) \to \lambda\,(y) \to \lambda\,(z) \to x+y+z \\ f\,(x,y,z) & = & x+y+z \end{array}$$

That's much easier to read. It should be understood that the parentheses are right-associative: the parentheses "associate" rightward — i.e. it's $a \to (b \to (c \to d))$, not $((a \to b) \to c) \to d$.[16]

That's Currying for you.

4.1.1 Piecewise functions

As a random aside, I'm going to introduce you to the *piecewise function*. It's a function whose definition changes based on the input.

$$\begin{array}{rcl} q & : & \mathbb{N} \to \mathbb{Z} \\ q(x) & := & \begin{cases} x \text{ is even} & \to & \frac{x}{2} \\ x \text{ is odd} & \to & \text{`}\left(\frac{x+1}{2}\right) \end{cases} \end{array}$$

Let's look at $q(0)$: 0 is even, so $q(0) = \frac{0}{2} = 0$.

Let's make a table:

x	$q(x)$	$q(x)$ reduced
0	$0 \div 2$	0
1	$\text{`}((1+1) \div 2)$	`1
2	$2 \div 2$	1
3	$\text{`}((3+1) \div 2)$	`2
4	$4 \div 2$	2
5	$\text{`}((5+1) \div 2)$	`3
6	$6 \div 2$	3
7	$\text{`}((7+1) \div 2)$	`4
8	$8 \div 2$	4

Hopefully you get this. It's pretty simple.

4.1.2 Vocabulary

I've been sort of dropping these vocabulary terms throughout the beginning of the chapter. That said, I'll list them here, so you know where they are. (They're also in § A.5).

1. All functions are *transparent* — $a = b \implies f(a) = f(b)$
2. If $f : A \to B$, then A is the *domain* of f and B is the *codomain* of f.

3. If $f : A \to B$, and there are no two distinct elements of A that map to the same thing in B, then f is *injective*.

$$f : A \to B$$
$$\nexists\,(a,b)\,;\; a,b \in A \wedge a \neq b \wedge f(a) = f(b) \iff f \text{ is injective}$$

4. If $f : A \to B$, then the elements in B that can be expressed as $f(x)\,;\; x \in A$ form the *image*.

$$f : A \to B$$
$$\operatorname{im}(f) = \{\, f(x) \in B;\; x \in A \,\}$$

5. If the image of a function is equal to its codomain, then the function is *surjective*.

$$f : A \to B$$
$$B = \{\, f(x) \in B;\; x \in A \,\} \iff f \text{ is surjective}$$

6. If a function is both injective and surjective, then it is *bijective*.

7. Some functions have *inverses*. That is, if

$$f : A \to B$$

$$\operatorname{arc}(f) : B \to A$$
$$\operatorname{arc}(f,x) = x;\; \forall x \in A$$

Remember that, because of currying, $\operatorname{arc}(f,x) = \operatorname{arc}(f)(x)$. That is:

$$\begin{aligned} f &: A \to B \\ \operatorname{arc} &: (A \to B) \to B \to A \\ \operatorname{arc}(f) &: B \to A \end{aligned}$$

If a function has an inverse, it is said to be *invertible*.

8. If a function is invertible, then the image of the inverse is called the *preimage*.

4.1.3 Exercises

Ex. 7 — I knew you were going to just gloss over those, so I made a really hard (i.e. fun) problem: prove that a function is invertible if (and only if) it is bijective. This is a very difficult proof, but you really need to understand it.

Chapter 5

More stuff about sets (and functions)

By now, you hopefully have some idea into the basic intuition behind sets and functions. Moreover, you've proven some cool stuff about them – for instance, you proved that a function is invertible iff it is bijective.

That's kind of cool, right? It's much easier to verify that a function is bijective than it is to find its inverse. So, right off the bat, you can see if a function is invertible without trying to invert it. Despite what you may think, a common problem in math is to find inverse functions.

Speaking of functions, I'm going to define a really simple function:

$$\begin{aligned} &\mathrm{id} : a \to a \\ &\mathrm{id}\,(x) = x \end{aligned}$$

That function is about as simple as functions get. It's not a very interesting function, but it's handy when defining things.

This is a slightly less simple function, but nonetheless important

$$\begin{aligned} &\mathrm{flip} : (a \to b \to c) \to b \to a \to c \\ &\mathrm{flip}\,(f,x,y) = f\,(y,x) \end{aligned}$$

It takes a function, f, which takes an a and a b, and then returns another function with the arguments flipped. You won't usually see $\text{flip}\,(\lambda\,(x,y)\to x-y,3,5)$ floating around. Instead

$$\begin{aligned} f &: p\to q\to r \\ \text{flip}\,(f) &: q\to p\to r \end{aligned}$$

Here's an infix operator:

$$\begin{aligned} &\circ : (b\to c)\to(a\to b)\to a\to c \\ &(f\circ g)\,(x) = f\,(g\,(x)) \end{aligned}$$

We're going to look at some more stuff with sets.

5.1 Set subtraction

First off, we have "set subtraction". This is sometimes called the "relative complement".

$$P\setminus Q = \{\,x\in P;\; x\notin Q\,\}$$

$P\setminus Q$ is all of the elements in P that are not in Q.

What would $P\setminus P$ be, then? Well, $\varnothing$, of course, right!

$$P\setminus P = \{\,x\in P;\; x\notin P\,\} = \varnothing$$

Well, that looks like a contradiction, doesn't it? Well, sort of. It's a contradiction if you assume that P is nonempty — if you assume that there is some element in P — if an element is both in P and not in P, that would be madness! But, if you realize that the comprehension is the "set of objects satisfying the condition",

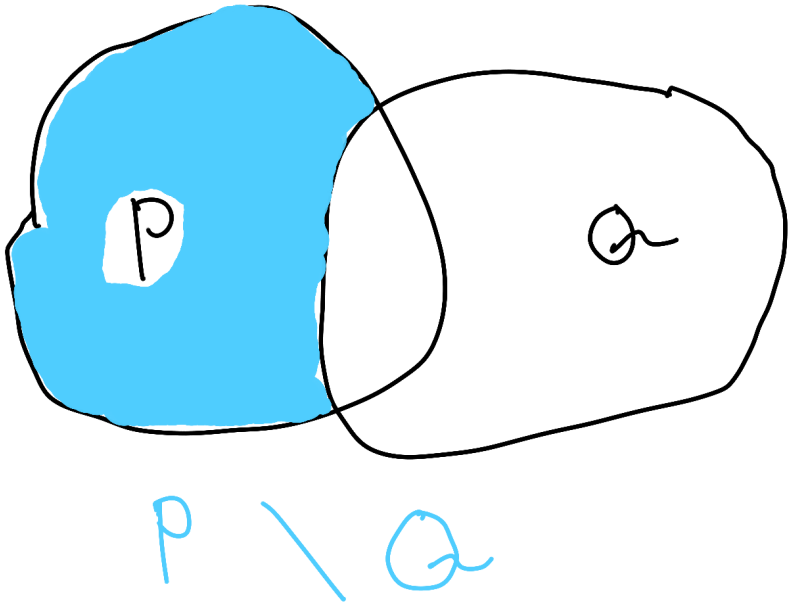


Figure 5.1: Set subtraction

then you don't encounter a contradiction. $P \setminus P$ is the set of all objects in P that are not in P: there are no elements satisfying this condition, thus $P \setminus P \equiv \varnothing$.

Figure 5.1 explains this idea graphically.

5.1.1 Complement

If $P \subset \mathcal{A}$, then $\mathcal{A} \setminus P$ is called the "*complement* of P with respect to $\mathcal{A}$". Yes, that's compl*e*ment with an 'e', not compl*i*ment, with an 'i'.

I drew another diagram to illustrate the complement in fig. 5.2.

In these exercises, you are going to prove every identity there is about sets.

5.1.2 Exercises

Ex. 8 — $A \setminus (B \cap C) \equiv (A \setminus B) \cup (A \setminus C)$

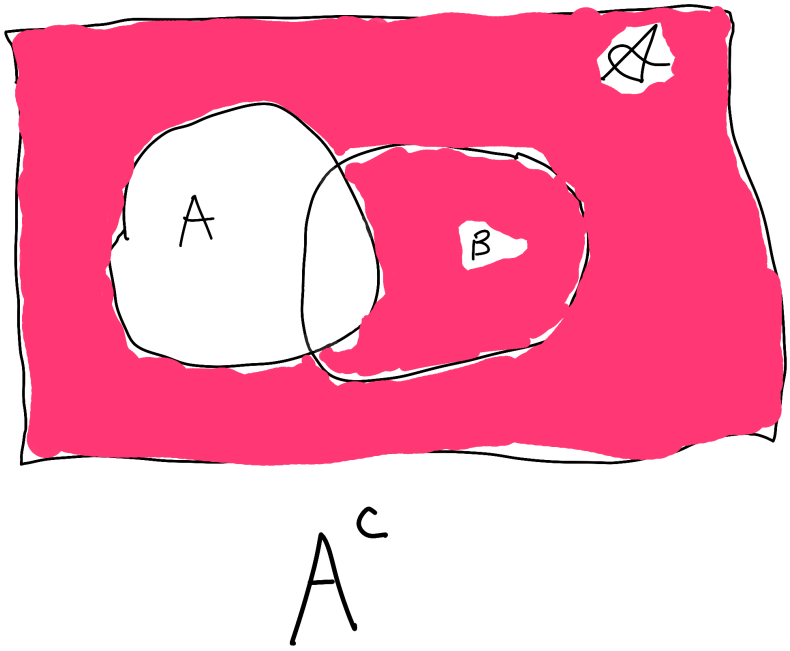


Figure 5.2: The complement, illustrated graphically. I tried to draw an $\mathcal{A}$ in the upper right corner. It turned out terribly.

Ex. 9 — $A \setminus (B \cup C) \equiv (A \setminus B) \cap (A \setminus C)$

Ex. 10 — $A \setminus (B \setminus C) \equiv (A \setminus B) \cup (A \cap C)$

Ex. 11 — $(A \setminus B) \cap C \equiv (A \cap C) \setminus B \equiv A \cap (C \setminus B)$

Ex. 12 — $(A \setminus B) \cup C \equiv (A \cup C) \setminus (B \setminus C)$

Ex. 13 — $A \setminus A \equiv \varnothing$

Ex. 14 — $A \setminus \varnothing \equiv A$

Ex. 15 — $\varnothing \setminus A \equiv \varnothing$

Ex. 16 — $A \cap (B \cup C) \equiv (A \cap B) \cup (A \cap C)$

Ex. 17 — $A \cup (B \cap C) \equiv (A \cup B) \cap (A \cup C)$

Ex. 18 — $(A^c)^c \equiv A$

Ex. 19 — $(A \cap B)^c \equiv A^c \cup B^c$

Ex. 20 — $(A \cup B)^c \equiv A^c \cap B^c$

5.2 Cartesian products

The next thing we need to go over is a *Cartesian product* - it's basically a way to double up on a set. So, say we have a set A, and another set B, the Cartesian product is the set of all 2-vectors, where the first element is from A, and the second element is from B.

$$\begin{aligned} &\times : \mathbf{Set}(a) \to \mathbf{Set}(b) \to \mathbf{Class}(a,b) \\ &A \times B := \{ (x,y) ; x \in A \wedge y \in B \} \end{aligned}$$

It's the class of all pairs of the elements in either set. It is actually a set, too, but I haven't taught you enough to prove that. I also haven't taught you what a class is. So, for the time being, know that the Cartesian product produces a set, but at the same time you don't know that.

Let's see some examples!

$$\begin{aligned} \{1,2,3\} \times \{4,5,6\} \quad = \quad \{ \quad & (1,4) \\ , \quad & (1,5) \\ , \quad & (1,6) \\ , \quad & (2,4) \\ , \quad & (2,5) \\ , \quad & (2,6) \\ , \quad & (3,4) \\ , \quad & (3,5) \\ , \quad & (3,6) \\ \} \quad & \end{aligned}$$

Alright, remember earlier when I told you to install Haskell? Well, if you didn't, do it now: **??**.

```
% ghci
GHCi, version 7.8.4: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
Prelude>
```

Listing 5.1:

GHCi is the Glasgow Haskell Compiler, interactive. It's an interactive interpreter for Haskell.

If you want to change your prompt to something other than `Prelude>`, then write something like this:

```
Prelude> :set prompt "ghci: "
ghci:
```

Listing 5.2:

If you want to do this permanently, add `:set prompt "ghci:  "` to `~/.ghc/ghci.conf`.

I didn't just have you open up GHCi for no good reason. It's really easy to play with these Cartesian products in GHCi. This way, you can experiment.

Remember the definition of the Cartesian product:

$$A \times B := \{ (x,y) ; x \in A \wedge y \in B \}$$

Let's try that example out. Math doesn't directly translate into Haskell. Some of the syntax is a bit different.

```
ghci: [(a,b) | a <- [1,2,3], b <- [4,5,6]]
[(1,4),(1,5),(1,6),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6)]
it :: (Num t1, Num t) => [(t, t1)]
```

Listing 5.3:

That last line probably doesn't show up for you. It's just telling us the type of our expression. To have it show up automatically for you, run `:set +t`, or add it to `~/.ghc/ghci.conf`.

There are a number of ways to actually work out the mechanics of the product. The simplest, and easiest way, is through *recursion*. So, let's go over the mechanics of $\{1,2,3\} \times \{4,5,6\}$. You take the first element of the first set, in this case, 1, and take the Cartesian product of $\{1\}$ and $\{4,5,6\}$:

$$\{1\} \times \{4,5,6\} = \{(1,4),(1,5),(1,6)\}$$

That's unreadable

$$\begin{array}{rcl} \{1\} \times \{4,5,6\} & = & \{ \quad (1,4) \\ & & , \quad (1,5) \\ & & , \quad (1,6) \\ & & \} \end{array}$$

Pretty easy. Then you do the same thing with the second element.

$$\begin{array}{rcl} \{2\} \times \{4,5,6\} & = & \{ \quad (2,4) \\ & & , \quad (2,5) \\ & & , \quad (2,6) \\ & & \} \end{array}$$

You guessed it!

$$\begin{array}{rcl} \{3\}\times\{4,5,6\} & = & \{\ (3,4) \\ & & ,\ (3,5) \\ & & ,\ (3,6) \\ & & \} \end{array}$$

Then you take the union:

$$\begin{array}{rcl} \{1,2,3\}\times\{4,5,6\} & = & \bigcup\ \{\ \{1\}\times\{4,5,6\}, \\ & & ,\ \{2\}\times\{4,5,6\}, \\ & & ,\ \{3\}\times\{4,5,6\}, \\ & & \} \end{array}$$

I'm going to have to display the intermediate result in a separate thing, because it's just awful if I try to smush it in with the rest:

$$\begin{array}{l} \{\ (1,4) \\ ,\ (1,5) \\ ,\ (1,6) \\ \} \end{array} \bigcup \begin{array}{l} \{\ (2,4) \\ ,\ (2,5) \\ ,\ (2,6) \\ \} \end{array} \bigcup \begin{array}{l} \{\ (3,4) \\ ,\ (3,5) \\ ,\ (3,6) \\ \} \end{array}$$

Which evaluates to

$$\begin{array}{rcl} \{1,2,3\}\times\{4,5,6\} & = & \{\ (1,4) \\ & & ,\ (1,5) \\ & & ,\ (1,6) \\ & & ,\ (2,4) \\ & & ,\ (2,5) \\ & & ,\ (2,6) \\ & & ,\ (3,4) \\ & & ,\ (3,5) \\ & & ,\ (3,6) \\ & & \} \end{array}$$

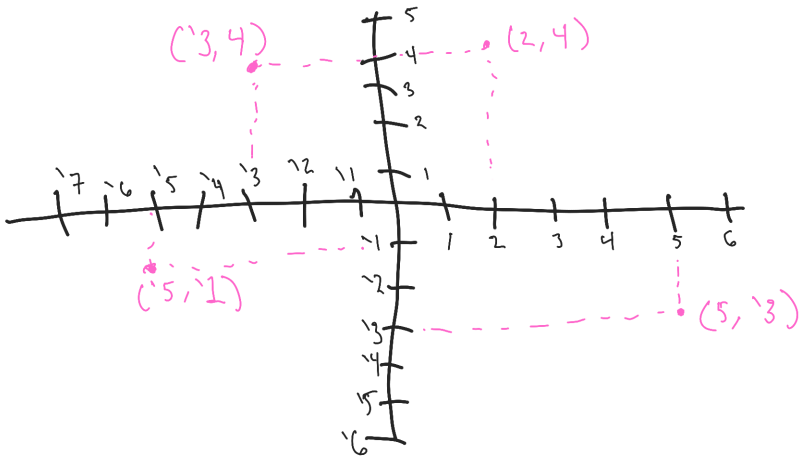
Okay, great. I hope you understand this so far. What happens when we take the product of a set with itself?

$$\begin{aligned}\{1,2,3\}\times\{1,2,3\} = \{ & (1,1)\\ , & (1,2)\\ , & (1,3)\\ , & (2,1)\\ , & (2,2)\\ , & (2,3)\\ , & (3,1)\\ , & (3,2)\\ , & (3,3)\\ \}\end{aligned}$$

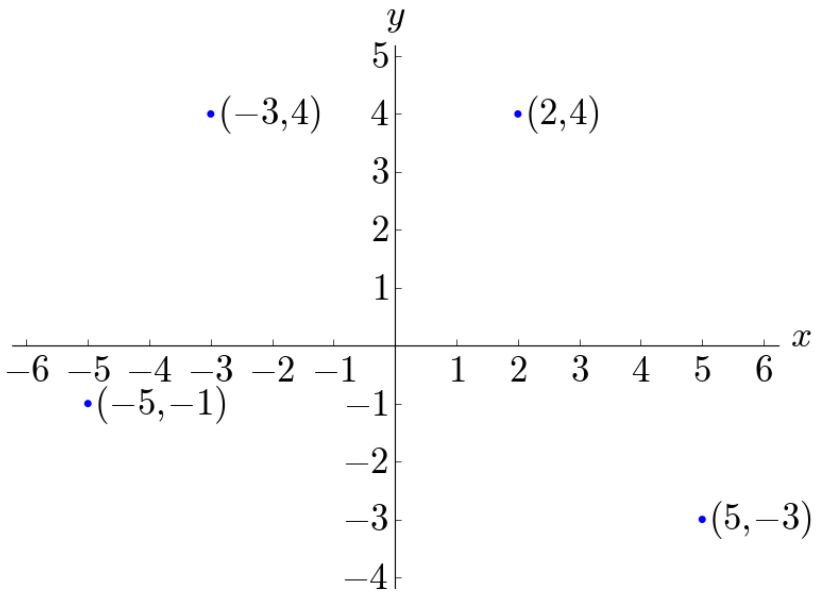
The Cartesian product of a set A with itself is usually denoted A^2 instead of $A \times A$. Like I said, we like to be lazy.

5.3 Function plots

Okay, so, here's something that doesn't really fit in anywhere else. Now that you know what Cartesian products are, as well as vectors, I can introduce you to the *Cartesian coordinate plane*. Basically, it's a graphical representation of $\mathbb{R} \times \mathbb{R}$:



I also put some vectors on there. That looks really crappy, let me draw that with a computer real quick:



The source code for that graph is in listing B.1.

As far as conventions go, you always list the horizontal coordinate first, and the vertical coordinate second. Usually, the horizontal axis is labeled the x-axis, and the vertical axis is labeled the y-axis. Hence, the convention is to denote vectors on the plane as (x,y)

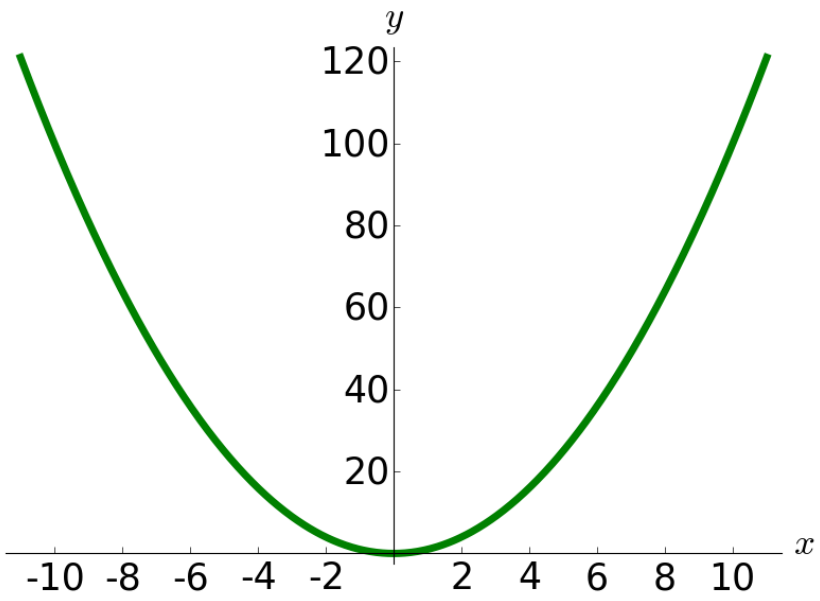
Okay, so given $A \times B$, a point (x,y) on the Cartesian coordinate plane represents the vector (x,y), where $x \in A$, $y \in B$. With $\mathbb{R}$, a point (x,y) on the plane is the vector $(x,y) \in \mathbb{R}^2$

So, how do we plot a function? Well, basically, given a function $f : A \to B$, you plot $(x, f(x)) \in A \times B$.

You plot the domain on the x-axis, and the codomain on the y-axis. You put the input value as the horizontal coordinate, and the output value as the y-coordinate.

For instance:

$$\left(\lambda\left(x\right) \to x^2\right) : \mathbb{R} \to \mathbb{R}$$



The source is in listing B.2.

Go to some point on the x-axis. Then go upwards from there until you hit the line. Let's start with 8. Find 8 on the x-axis, then go up from there until you get to the line. If you look, the vertical coordinate of the line at that point is 64, which is the square of 8. That's kind of cool.

5.4 The work of Georg Cantor

Without further ado, we're going to look at Georg Cantor's work. Georg Cantor, if you remember from § 3, is the guy who first studied sets. He came up with some bizzarre results.

First of all, long before Cantor, another guy, of whom you've likely heard, Galileo Galilei, came up with a paradox. Galileo Galilei is usually mononymously referred to as Galileo, mostly because it's easier to type. Galileo is most

famous for championing the idea that the earth revolves around the sun, and not the other way around. He spent his last days under house arrest because he believed that. Seventeenth century Italy didn't really have the same free speech protections found today in the first world.

Anyway, I digress. Aside from his amazing work in physics, Galileo was among the first to point out an interesting fact about $\mathbb{N}$: there are as many perfect squares as there are natural numbers.[17] That's weird, because the perfect squares are a subset of the natural numbers.

$$\begin{array}{ccccccc} 1 & 2 & 3 & 4 & \dots & n & \dots \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ 1 & 4 & 9 & 16 & \dots & n^2 & \dots \end{array}$$

To be fair, Galileo wasn't the first person to come up with this paradox, he was just among the first. He was the most famous person to come up with this paradox, hence why it's named after him.[8]

So, back to the real world: despite being a subset, the infinite set of perfect squares has as many elements as the infinite superset of natural numbers. Galileo decided that the only solution was to not consider words like "larger" or "smaller" when discussing infinite sets. Eventually, mathematicians started talking about "larger" and "smaller" in the context of infinite sets, just not the way Galileo would have.

If you want to look at this another way, we've found a bijection

$$\begin{aligned} &f : \mathbb{N} \to \left\{ x \in \mathbb{N};\ y \in \mathbb{N}, x = y^2 \right\} \\ &f(x) := x^2 \end{aligned}$$

Galileo, and later Cantor, decided that two sets have the same number of elements if there exists a bijection between them. Instead of saying "have the same number of elements", we instead say "have the same cardinality".

So, $\{1,2,3\}$ has the same cardinality as $\{4,5,6\}$, because there's a bijective relation between them.

$$(\lambda\,(x) \to x+3) : \{\,1,2,3\,\} \to \{\,4,5,6\,\}$$

If two sets are finite, as is the case with the previous example, their cardinality is just the number of elements. So, the cardinality of $\{\,1,2,3\,\}$ is just 3. If two sets A and B have the same cardinality, then I'm going to write $A \overset{c}{=} B$, which you should read as "A is cardinally equal to B".

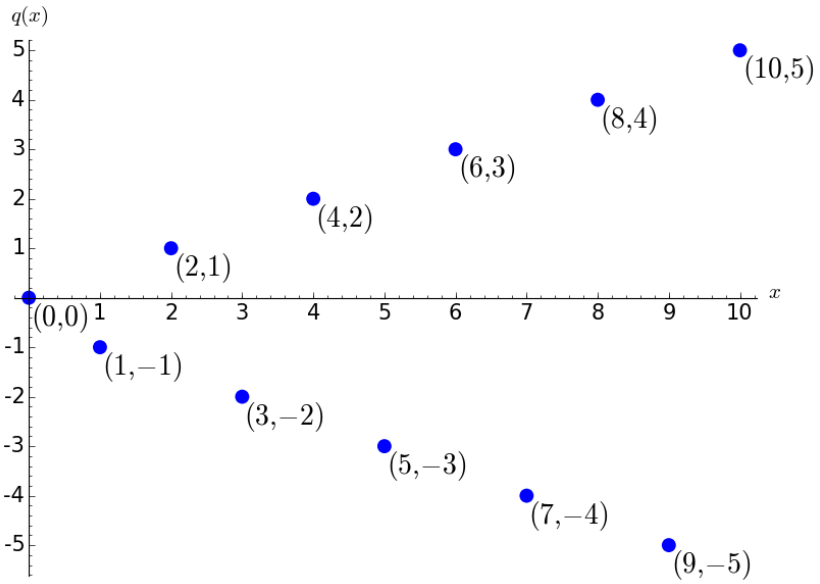
Do you remember that example function from § 4.1.1?

$$\begin{array}{rcl} q & : & \mathbb{N} \to \mathbb{Z} \\ q\,(x) & := & \begin{cases} x \text{ is even} & \to & \frac{x}{2} \\ x \text{ is odd} & \to & \text{`}\left(\frac{x+1}{2}\right) \end{cases} \end{array}$$

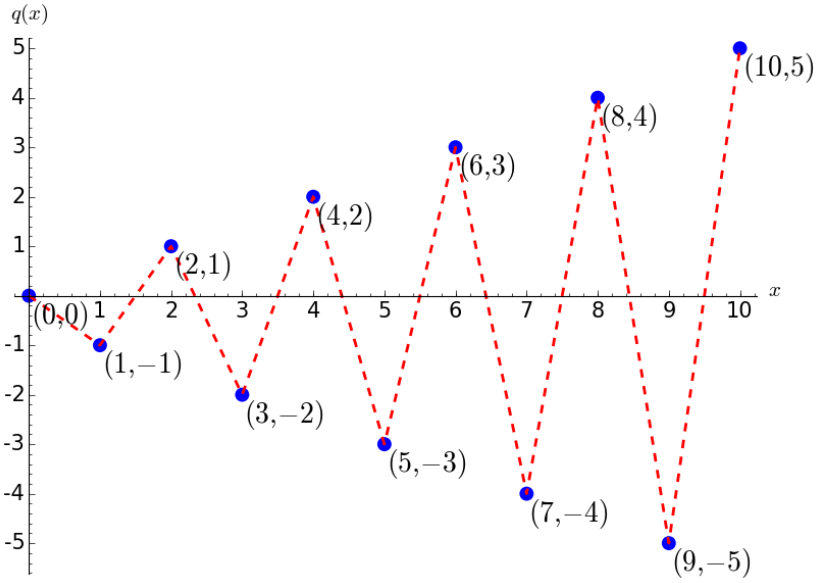
x	$q\,(x)$	$q\,(x)$ reduced
0	$0 \div 2$	0
1	$\text{`}((1+1) \div 2)$	`1
2	$2 \div 2$	1
3	$\text{`}((3+1) \div 2)$	`2
4	$4 \div 2$	2
5	$\text{`}((5+1) \div 2)$	`3
6	$6 \div 2$	3
7	$\text{`}((7+1) \div 2)$	`4
8	$8 \div 2$	4

Wait wait wait!!!!! That's a bijection! Holy crap! So $\mathbb{N} \overset{c}{=} \mathbb{Z}$! That's interesting! Again, despite $\mathbb{N} \subset \mathbb{Z}$, $\mathbb{N} \overset{c}{=} \mathbb{Z}$. That's kind of cool.

Let's plot that function:



Well that's a bit hard to follow. Let's draw a dotted line between each successive point:



Okay. Please note that since $q : \mathbb{N} \to \mathbb{Z}$, there aren't intermediate values. The

function only exists at the blue points. That is, you can't evaluate $q(2.5)$, because $2.5 \notin N$.

Anyway, the point is, we are able to enumerate through the values of $\mathbb{Z}$, the same way we can with $\mathbb{N}$. We've found a bijection $q : \mathbb{N} \to \mathbb{Z}$, therefore $\mathbb{N} = \mathbb{Z}$.

The cardinality of $\mathbb{N}$ (and also $\mathbb{Z}$) is called $\aleph_0$, pronounced "aleph-null". $\aleph$ is the first letter of the Hebrew alphabet, called "aleph".[1]

5.4.1 The cardinality of irrational numbers

What about $\mathbb{I}$? $\mathbb{I}$ is seemingly more infinite than $\mathbb{N}$. $\mathbb{N}$ and $\mathbb{Z}$ are discrete sets: it's pretty easy to enumerate through them. $\mathbb{I}$ is continuous though. Between any two values, there's always an infinity of more values.

Let's, for fun, try to list every single irrational number. If we can list every irrational number, then we must be able to enumerate through them, which would mean that $\mathbb{N} \overset{c}{=} \mathbb{I}$

[1] Some early math textbooks accidentally printed the $\aleph$ upside-down.[1] What a bunch of idiots.

```
1.714761022369152...
4.008668726427755...
1.566116992594829...
1.519257059116716...
5.011643808251281...
6.533800807168559...
9.838685190958348...
3.424290398329045...
5.065089480002634...
6.972994377235255...
7.763147189141261...
8.374868221801194...
2.901203914856270...
9.734153197637937...
1.163373088314136...
1.489918657733841...
1.775506328996835...
...
```

Listing 5.4:

I have to do it `in monospace` so that everything is aligned, sorry. Okay, let's take the first number, `1.714761022369152...`, and subtract 1 from its first digit: `0.714761022369152...`. Okay, easy enough

Let's do the same thing to the second number, with the second digit:

`4.008668726427755...` oops! It's a 0. Well, let's just make it cyclic - i.e. $0-1 \cong 9$.

So, we have `4.008668726427755...  -> 4.908668726427755...`

Let's list what we have so far

```
1.714761022369152... -> 0.714761022369152...
4.008668726427755... -> 4.908668726427755...
1.566116992594829...
1.519257059116716...
5.011643808251281...
6.533800807168559...
9.838685190958348...
3.424290398329045...
5.065089480002634...
6.972994377235255...
7.763147189141261...
8.374868221801194...
2.901203914856270...
9.734153197637937...
1.163373088314136...
1.489918657733841...
...

0.9
```

Listing 5.5:

So, for the *n*th number on the list, we change the *n*th digit. We're also going to take the output of the flipping process, and list it in a new number at bottom. Let's do this to a few more numbers, so you get the hang of it. I'm also going to add a space around the number I changed, to make it more obvious

```
 1  1 .714761022369152... ->  0 .714761022369152...
 2 4. 0 08668726427755... -> 4. 9 08668726427755...
 3 1.5 6 6116992594829... -> 1.5 5 6116992594829...
 4 1.51 9 257059116716... -> 1.51 8 257059116716...
 5 5.011 6 43808251281... -> 5.011 5 43808251281...
 6 6.5338 0 0807168559... -> 6.5338 9 0807168559...
 7 9.83868 5 190958348... ->
 8 3.424290 3 98329045... ->
 9 5.0650894 8 0002634... ->
10 6.97299437 7 235255... ->
11 7.763147189 1 41261... ->
12 8.3748682218 0 1194... ->
13 2.90120391485 6 270... ->
14 9.734153197637 9 37... ->
15 1.1633730883141 3 6... ->
16 ...
17 0.95849
```

Listing 5.6:

Okay, you're getting this! I'm sure you can figure out what the rest are:

```
 1   1.714761022369152... ->   0.714761022369152...
 2 4. 0 08668726427755... -> 4. 9 08668726427755...
 3 1.5 6 6116992594829... -> 1.5 5 6116992594829...
 4 1.51 9 257059116716... -> 1.51 8 257059116716...
 5 5.011 6 43808251281... -> 5.011 5 43808251281...
 6 6.5338 0 0807168559... -> 6.5338 9 0807168559...
 7 9.83868 5 190958348... -> 9.83868 4 190958348...
 8 3.424290 3 98329045... -> 3.424290 2 98329045...
 9 5.0650894 8 0002634... -> 5.0650894 7 0002634...
10 6.97299437 7 235255... -> 6.97299437 6 235255...
11 7.763147189 1 41261... -> 7.763147189 0 41261...
12 8.3748682218 0 1194... -> 8.3748682218 9 1194...
13 2.90120391485 6 270... -> 2.90120391485 5 270...
14 9.734153197637 9 37... -> 9.734153197637 8 37...
15 1.1633730883141 3 6... -> 1.1633730883141 2 6...
16 ...
17 0.95849427609582 ? ... -> 0.95849427609582 ? ...
```

Listing 5.7:

Wait wait wait! We've made a new irrational number that's different from all of the other numbers in at least 1 digit, right? It's different from the first number in its first digit, it's different from the second number in its second digit, and so on.

So, if we theoretically list all of the irrational numbers, in some sort of order, we can still make another irrational number that's different than each of them by at least one digit. Thus, it's impossible to list every single irrational number, because there's always another one!

You can make the same argument for $\mathbb{N}$, sort of. There's always another natural number, but it goes after all of the previous ones. With the irrational numbers, we can always make a new irrational number that goes somewhere in the middle of the set.

It's like if there was a long line to get tickets for a football game. Every second, there's a new person coming. With $\mathbb{N}$ and $\mathbb{Z}$, you can just stick the new person at the end of the line. With $\mathbb{I}$, however, you can't just stick the new person at the end, you have to put him at a definite spot in the middle.

When the ticket booth person needs to help the next person, it's impossible to determine who to help next, because every second there's a new person getting stuck in front of the person first in line. You can't take down the names of the first 20 people in line, because there's no concept of counting with the irrational numbers. That concept exists with the natural numbers. $\aleph_0$ is defined by being "countably infinite" - $\mathbb{I}$ is not countably infinite; it's its own type of infinite.

So $\mathbb{I} \overset{c}{>} \mathbb{N}$!

Because $\mathbb{R} \supset \mathbb{I}$, it's not possible that $\mathbb{R}$ is countably infinite. There's a part of $\mathbb{R}$ that you can't count, so you can't count all of $\mathbb{R}$. Pretty simple:

5.4.2 Rational numbers

Here's where it gets interesting. If you remember, $\mathbb{Q}$, the rational numbers, is all numbers that can be written as a ratio of $\frac{x}{y}$, where $x, y \in \mathbb{Z}, y \neq 0$. Incidentially, they are also numbers that follow some sort of pattern.

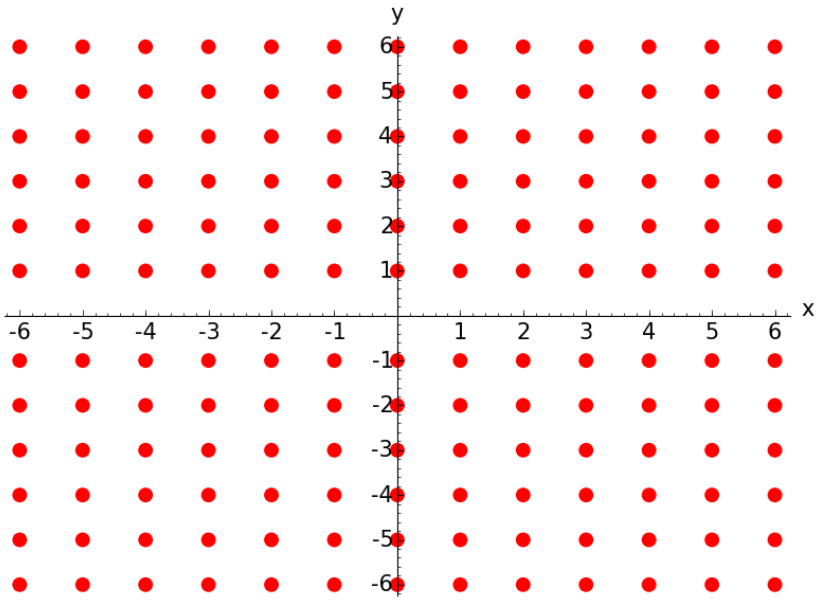
Well, that definition looks sort of familiar: What if we wrote $\mathbb{Q}$ this way:

$$\mathbb{Q} = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$$

Where $(x, y) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$ maps to the fraction $\frac{x}{y}$.

We've done Cartesian coordinate planes of $\mathbb{R}^2$ and $\mathbb{N} \times \mathbb{Z}$. Is there any reason we can't do one of $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$?

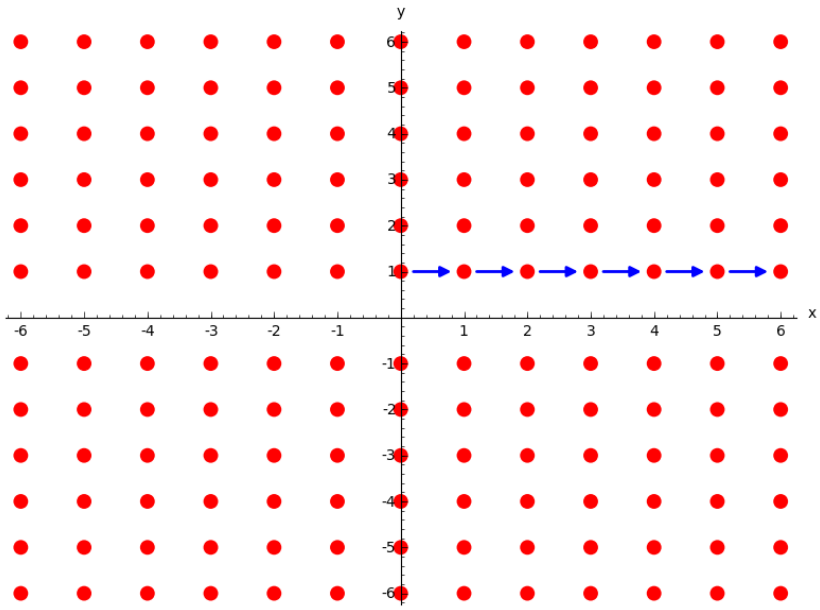
Well, of course not!



Every blue dot at some point (x, y) represents the fraction $\frac{x}{y}$. So, the dot at $(1, 3)$ represents the fraction $\frac{1}{3}$. Note that there are no points on the line $y = 0$, because you can't divide by zero.

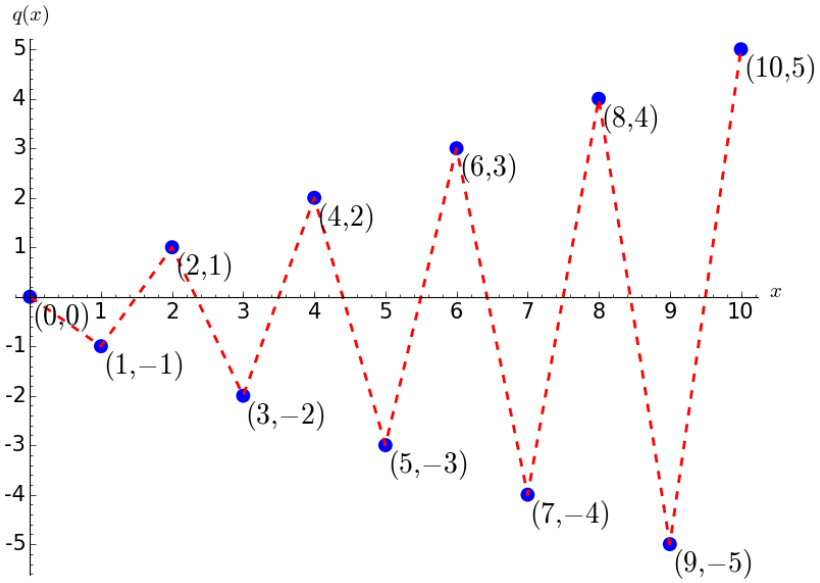
So, can we enumerate through those? That is, follow some pattern that will eventually encapsulate every rational number? There's no harm in trying!

The naïve way to do it would be to just head in one direction until you stop.

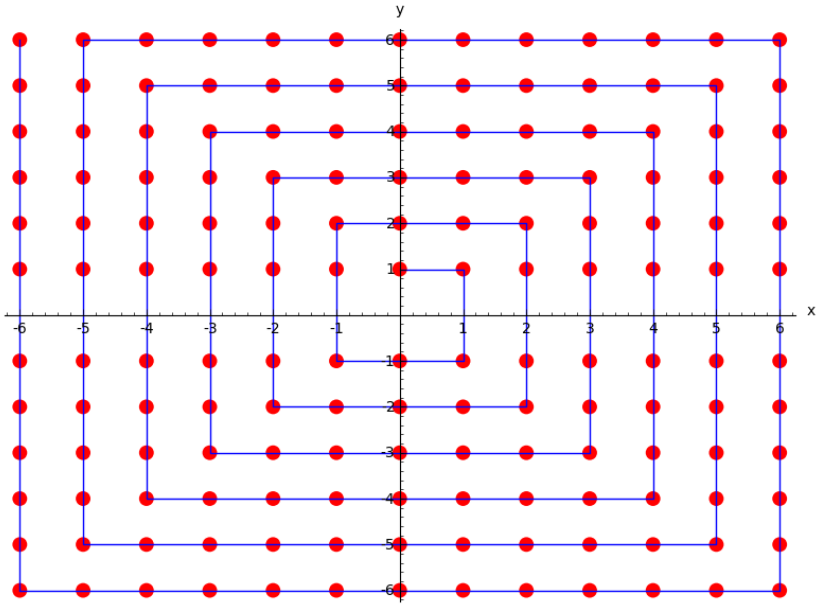


Of course, you never stop, so this won't work.

With the $\mathbb{N} \to \mathbb{Z}$ bijection, we sort of doubled back on ourselves:



Maybe let's try that with this:

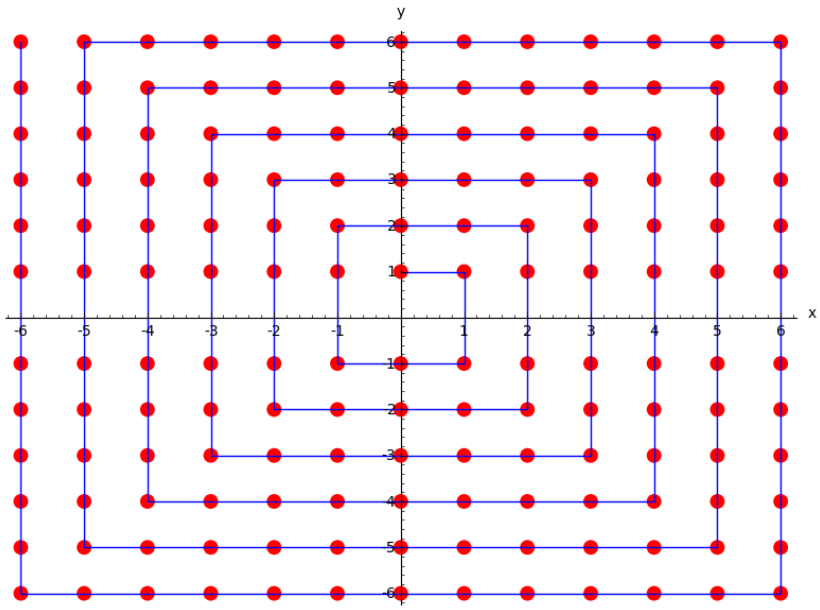


Hey! That works. If we keep going, we'll enumerate through $\mathbb{Q}$! So $\mathbb{Q} \stackrel{c}{=} \mathbb{N}$!

5.5 Okay, that's great. What the hell is that function?

Well, to be honest, I'm not sure. I wrote the script to generate the graph by doing some very shoddy programming, but I don't know the associated function. How about, as an, um, exercise, I explain my logic?

Let's look at that graph again:



Let's call the hypothetical function that defines that bijection r

$$r : \mathbb{N} \to \mathbb{Q}$$

First of all, we start at the coordinate $(0,1)$. So we can write

$$r(0) := (0,1)$$

5.6 Conclusion

These are the bizarre results Cantor found, which his colleagues refused to believe.

The method I used to prove that $\mathbb{I}$ was uncountable is called "Cantor's diagonal argument". He had proven years beforehand that $\mathbb{I} \overset{c}{>} \mathbb{N}$, using a completely different method. The diagonal method is much more approachable, so I used that instead.

More importantly, you can use the diagonal argument in a variety of different ways. The most interesting such way is Russell's paradox, which I'll get to in the next chapter.

The conclusion to draw from this section is:

> All infinite sets are infinite, but some are more infinite than others.
>
> – George Orwell

Appendices

Appendix A

Identities, theorems, and the like

This appendix just lists identities, theorems, and stuff like that. It's for reference, not for reading.

A.1 Equality

A.1.1 Properties

Reflexive property $a \equiv a$

Commutative property $(a = b) \iff (b = a)\,;\, \forall a, b$

Transitive property $(a = b) \wedge (b = c) \implies (a = c)\,;\, \forall a, b, c$

A.1.2 Notation

a = b means that a and b are the same thing.

a ≡ b means that $a = b$, for all a and b. $a \equiv b$ should be read "a is identically equivalent to b".

a := **b** means that a is defined to be equal to b. In practice, this is the same as $\equiv$, but is semantically different.

A.2 Implications

Reflexive property $a \implies a; \forall a$

Transitive property $(a \implies b) \wedge (b \implies c) \implies (a \implies c); \forall a,b,c$

Negation $(a \implies b) \iff (\neg a \impliedby \neg b); \forall a,b$

A.3 Booleans

Definition A *Boolean* is a value of either true or false. The study of Booleans is called *Boolean algebra*. The rules for Booleans also work for propositions. The set of Booleans is often referred to as $\mathbb{B} = \{\text{True}, \text{False}\}$

Logical-and $a \wedge b$ is pronounced "a logical-and b". It is true iff a and b are both true.

$$\wedge : \mathbb{B} \to \mathbb{B} \to \mathbb{B}$$

Logical-or $a \vee b$ is pronounced "a logical-or b". It is true if one or more of a and b are true.

$$\vee : \mathbb{B} \to \mathbb{B} \to \mathbb{B}$$

Logical-not $\neg a$ is pronounced "logical-not a". $\neg$ takes true to false, and false to true.

$$\neg : \mathbb{B} \to \mathbb{B}$$

Cancellative property $\neg \circ \neg \equiv \text{id}$

Nomenclature Booleans are named after George Boole, who was the first to study them to any extent.

A.3.1 Logical-and

Reflexive property $a \wedge a \equiv a$

Associative property $a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c$

Commutative property $a \wedge b \equiv b \wedge a$

Distributive property $a \wedge (b \vee c) \equiv (a \wedge b) \vee (b \wedge c)$

A.3.2 Logical-or

Reflexive property $a \vee a \equiv a$

Associative property $a \vee (b \vee c) \equiv (a \vee b) \vee c$

Commutative property $a \vee b \equiv b \vee a$

Distributive property $a \vee (b \wedge c) \equiv (a \vee b) \wedge (b \vee c)$

This is a consequence of the distributive property mentioned in § A.3.1, De Morgan's first law, and the cancellative property.

Proof. Start with the first property

$$a \wedge (b \vee c) \equiv (a \wedge b) \vee (b \wedge c)$$

Apply $\neg$ to both sides

$$\neg (a \wedge (b \vee c)) \equiv \neg ((a \wedge b) \vee (b \wedge c))$$

Apply De Morgan's laws

$$\neg a \vee \neg (b \vee c) \equiv \neg (a \wedge b) \wedge \neg (b \wedge c)$$

Do it again

$$\neg a \vee (\neg b \wedge \neg c) \equiv (\neg a \vee \neg b) \wedge (\neg b \vee \neg c)$$

Let $p, q, r = \neg a, \neg b, \neg c$, respectively.

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (q \vee r)$$

□

A.3.3 De Morgan's Laws

De Morgan's first law $\neg (a \wedge b) \equiv \neg a \vee \neg b$

Derived law $\neg (a \vee b) \equiv \neg a \wedge \neg b$

Proof. Start with the first law

$$\neg (a \wedge b) \equiv \neg a \vee \neg b$$

Let $p = \neg a$, $q = \neg b$

$$p \vee q \equiv \neg (\neg p \wedge \neg q)$$

Apply $\neg$ to both sides of $\equiv$

$$\neg (p \vee q) \equiv \neg p \wedge \neg q$$

□

A.4 Sets

A.4.1 Definitions

Unions $a \cup b := \{ x \in \mathcal{A};\ x \in a \vee x \in b \}$

Intersects $a \cap b := \{ x \in a;\ x \in b \}$

Set subtraction (or relative complement) $a \setminus b := \{ x \in a;\ x \notin b \}$

Complement (sometimes absolute complement) $a^c := \{ x \in \mathcal{A};\ x \notin a \}$

A.4.2 Identities

Unions

Reflexive property $a \cup a \equiv a$

Associative property $a \cup (b \cup c) \equiv (a \cup b) \cup c$

Commutative property $a \cup b \equiv b \cup a$

Intersects

Reflexive property $a \cap a \equiv a$

Associative property $a \cap (b \cap c) \equiv (a \cap b) \cap c$

Commutative property $a \cap b \equiv b \cap a$

Set subtraction identities

1. $\mathbf{A} \setminus (\mathbf{B} \cap \mathbf{C}) \equiv (\mathbf{A} \setminus \mathbf{B}) \cup (\mathbf{A} \setminus \mathbf{C})$

   Proof. Let $A, B, C \subseteq \mathcal{A}$.

$$\begin{aligned} A \setminus (B \cap C) &:= \{x \in A;\ x \notin \{y \in B;\ y \in C\}\} \\ &:= \{x \in A;\ x \notin B \vee y \notin C\} \end{aligned}$$

$$\begin{aligned} (A \setminus B) \cup (A \setminus C) &:= \{x \in \mathcal{A};\ x \in \{y \in A;\ y \notin B\} \vee x \in \{z \in A;\ z \notin C\}\} \\ &:= \{x \in \mathcal{A};\ (x \in A \wedge x \notin B) \vee (x \in A \wedge x \notin C)\} \end{aligned}$$

$$x \in A \implies x \in \mathcal{A}$$

Therefore

$$\begin{array}{rcl}(A \setminus B) \cup (A \setminus C) & := & \{x \in A;\, x \notin B \vee x \notin C\} \\ A \setminus (B \cap C) & := & \{x \in A;\, x \notin B \vee y \notin C\} \\ A \setminus (B \cap C) & \equiv & (A \setminus B) \cup (A \setminus C)\end{array}$$

□

2. $\mathbf{A} \setminus (\mathbf{B} \cup \mathbf{C}) \equiv (\mathbf{A} \setminus \mathbf{B}) \cap (\mathbf{A} \setminus \mathbf{C})$

Proof.

$$\begin{array}{rcl}A \setminus (B \cup C) & := & \{x \in A;\, x \notin B \wedge x \notin C\} \\ (A \setminus B) \cap (A \setminus C) & := & \{x \in \mathcal{A};\, x \in \{y \in A;\, y \notin B\} \wedge x \in \{z \in A;\, z \notin C\}\} \\ & := & \{x \in A;\, x \notin B \wedge x \notin C\}\end{array}$$

□

3. $\mathbf{A} \setminus (\mathbf{B} \setminus \mathbf{C}) \equiv (\mathbf{A} \setminus \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C})$

Proof.

$$\begin{array}{rcl}A \setminus (B \setminus C) & := & \{x \in A;\, x \notin \{y \in B;\, y \notin C\}\} \\ & := & \{x \in A;\, x \notin B \vee x \in C\} \\ (A \setminus B) \cup (A \cap C) & := & \{x \in \mathcal{A};\, x \in \{y \in A;\, y \notin B\} \vee x \in \{z \in A;\, z \in C\}\} \\ & := & \{x \in A;\, x \notin B \vee x \in C\}\end{array}$$

□

4. $(\mathbf{A} \setminus \mathbf{B}) \cap \mathbf{C} \equiv (\mathbf{A} \cap \mathbf{C}) \setminus \mathbf{B} \equiv \mathbf{A} \cap (\mathbf{C} \setminus \mathbf{B})$

Proof.

$$\begin{array}{rcl}(A \setminus B) \cap C & := & \{x \in \mathcal{A};\, x \in \{y \in A;\, y \notin B\} \wedge x \in C\} \\ & := & \{x \in A;\, x \notin B \wedge x \in C\} \\ (A \cap C) \setminus B & := & \{x \in \mathcal{A};\, x \in \{y \in A;\, y \in C\} \wedge x \notin B\} \\ & := & \{x \in A;\, x \notin B \wedge x \in C\} \\ A \cap (C \setminus B) & := & \{x \in A;\, x \notin B \wedge x \in C\}\end{array}$$

□

5. $(\mathbf{A} \setminus \mathbf{B}) \cup \mathbf{C} \equiv (\mathbf{A} \cup \mathbf{C}) \setminus (\mathbf{B} \setminus \mathbf{C})$

Proof.

$$\begin{array}{rcl}
(A \setminus B) \cup C & := & \{ x \in \mathcal{A};\ x \in \{ y \in A;\ y \notin B \} \vee x \in C \} \\
& := & \{ x \in \mathcal{A};\ (x \in A \wedge x \notin B) \vee x \in C \} \\
(A \cup C) \setminus (B \setminus C) & := & \{ x \in \mathcal{A};\ x \in \{ y \in \mathcal{A};\ y \in A \vee y \in C \} \wedge x \notin \{ z \in B;\ z \notin C \} \} \\
& := & \{ x \in \mathcal{A};\ (x \in A \vee x \in C) \wedge \neg (x \in B \wedge x \notin C) \} \\
& := & \{ x \in \mathcal{A};\ (x \in A \vee x \in C) \wedge (x \notin B \vee x \in C) \} \\
& := & \{ x \in \mathcal{A};\ x \in C \vee (x \in A \wedge x \notin B) \}
\end{array}$$

□

6. $\mathbf{A} \setminus \mathbf{A} \equiv \varnothing$

Proof.

$$A \setminus A := \{ x \in A;\ x \notin A \}$$

There are no elements in A that are also not in A, and the set with no elements is $\varnothing$. □

7. $\mathbf{A} \setminus \varnothing \equiv \mathbf{A}$

Proof.

$$A \setminus \varnothing := \{ x \in A \setminus x \notin \varnothing \}$$

$\varnothing$, by definition has no elements, so all elements in A satisfy the condition $x \notin \varnothing$. Thus,

$$A \setminus \varnothing \equiv A$$

□

8. $\varnothing \setminus \mathbf{A} \equiv \varnothing$

Proof.

$$\varnothing \setminus A := \{ x \in \varnothing;\ x \notin A \}$$

There are no elements in $\varnothing$, so everything fails the condition on the left-hand-side of the ; , hence $\varnothing \setminus A \equiv \varnothing$. □

Distributive properties

$\mathbf{A} \cap (\mathbf{B} \cup \mathbf{C}) \equiv (\mathbf{A} \cap \mathbf{B}) \cup (\mathbf{A} \cap \mathbf{C})$

Proof.

$$\begin{array}{rcl} A \cap (B \cup C) & := & \{ x \in A;\ x \in B \vee x \in C \} \\ (A \cap B) \cup (A \cap C) & := & \{ x \in \mathcal{A};\ (x \in A \wedge x \in B) \vee (x \in A \wedge x \in C) \} \\ & := & \{ x \in \mathcal{A};\ x \in A \wedge (x \in B \vee x \in C) \} \\ & := & \{ x \in A;\ x \in B \vee x \in C \} \end{array}$$

□

$\mathbf{A} \cup (\mathbf{B} \cap \mathbf{C}) \equiv (\mathbf{A} \cup \mathbf{B}) \cap (\mathbf{A} \cup \mathbf{C})$

Proof.

$$\begin{array}{rcl} A \cup (B \cap C) & := & \{ x \in \mathcal{A};\ x \in A \vee (x \in B \wedge x \in C) \} \\ (A \cup B) \cap (A \cup C) & := & \{ x \in \mathcal{A};\ (x \in A \vee x \in C) \wedge (x \in A \vee x \in C) \} \\ & := & \{ x \in \mathcal{A};\ x \in A \vee (x \in B \wedge x \in C) \} \end{array}$$

□

Complements

$(\mathbf{A}^{\mathbf{c}})^{\mathbf{c}} \equiv \mathbf{A}$

Proof.

$$\begin{array}{rcl} A \setminus (A \setminus B) & \equiv & (A \setminus A) \cup (A \cap B) \\ & \equiv & \varnothing \cup (A \cap B) \\ & \equiv & A \cap B \\ (A^c)^c & := & \mathcal{A} \setminus (\mathcal{A} \setminus A) \\ & := & \mathcal{A} \cap A \\ & := & A \end{array}$$

□

De Morgan's law $(\mathbf{A} \cap \mathbf{B})^{\mathbf{c}} \equiv \mathbf{A}^{\mathbf{c}} \cup \mathbf{B}^{\mathbf{c}}$

Proof.

$$\begin{aligned} A \setminus (B \cap C) &\equiv (A \setminus B) \cup (A \setminus C) \\ \mathcal{A} \setminus (A \cup B) &\equiv (\mathcal{A} \setminus A) \cup (\mathcal{A} \setminus B) \\ &\equiv A^c \cup B^c \end{aligned}$$

□

De Morgan's derived law $(\mathbf{A} \cup \mathbf{B})^{\mathbf{c}} \equiv \mathbf{A}^{\mathbf{c}} \cap \mathbf{B}^{\mathbf{c}}$

Proof.

$$\begin{aligned} A \setminus (B \cup C) &\equiv (A \setminus B) \cap (A \setminus C) \\ \mathcal{A} \setminus (A \cup B) &\equiv (\mathcal{A} \setminus A) \cap (\mathcal{A} \setminus B) \\ &\equiv A^c \cap B^c \end{aligned}$$

□

A.4.3 ZFC

ZFC Short for Zermelo-Fraenkel-Choice: a set of axioms rigorously describing set theory.

Nomenclature Named after Ernst Zermelo, who formulated the axioms, and Abraham Fraenkel, who greatly improved them.

Russell's paradox A paradox proposed by Bertrand Russell in the early 20$^{\text{th}}$ century regarding unrestricted set comprehensions

$$\begin{aligned} A &= \{x;\, x \notin x\} \\ A &\overset{?}{\in} A \end{aligned}$$

ZF ZFC without the axiom of choice

A.5 Functions

A.5.1 Vocabulary

Function A mathematical construct mapping an input to an output.

Referential transparency $a = b \implies f(a) = f(b)$. All functions are referentially transparent.

Domain If $f : A \to B$, then A is the *domain* of f.

Codomain If $f : A \to B$, then B is the *codomain* of f.

Image $\operatorname{im}(f) := \{ f(x) \in B;\ x \in A \}$

Injectivity $\nexists (a,b);\ a,b \in A \wedge a \neq b \wedge f(a) = f(b)$

Surjectivity $\mathbf{codom}(f) = \operatorname{im}(f)$

Bijectivity A function is *bijective* if it is both injective and surjective.

Invertibility A function is invertible iff it is bijective. The inverse of f is $\operatorname{arc}(f)$

Preimage $\operatorname{preim} := \mathbf{codom} \circ \operatorname{arc}$

Argument The specific input values to a function.

Signature If $f : A \to B$ is a function, then $A \to B$ is its signature.

A.5.2 Notation

: notation $f : A \to B$ means that f takes an item from A, and outputs an item to B, where A and B are types.

Currying Taking a function of multiple arguments, and transforming it into a chain of functions each taking one argument.

Normal signature

$+ : (\mathbb{C}, \mathbb{C}) \to \mathbb{C}$

Curried signature:

$+ : \mathbb{C} \to \mathbb{C} \to \mathbb{C}$

This doesn't change the behavior of the function, only the semantics.

Likewise, *uncurrying* is to undo the currying.

Composition We can smush two functions together with $\circ$:

$$\circ : (b \to c) \to (a \to b) \to a \to c$$
$$(f \circ g)(x) := f(g(x))$$

A.6 Lambda calculus

λ **abstraction** A way to write a function: $\lambda(x,y) \to x+y$

α **conversion** Changing the names of the arguments. For instance, you can write the above function as

$$\lambda(a,b) \to a+b$$

β **reduction** Partially calculating a result. For instance

$$\lambda(2,y) \to 2+y$$

Can be β reduced to

$$\lambda(y) \to 2+y$$

η **conversion** Removing or adding extraneous free arguments. The last function

$$\lambda(2,y) \to 2+y$$

Can be η *reduced* to

$$2+$$

Which could then be η *abstracted* to

$$\lambda(2,\kappa) \to 2+\kappa$$

A.7 Greek alphabet

Letter	Pronounciation	Rough latin equivalent
A, α	Alpha	A
B, β	Beta	B
Γ, γ	Gamma	G
Δ, δ	Delta	D
E, ε	Epsilon	E, j**e**t, phl**eg**m
Z, ζ	Zeta	Z
H, η	Eta	Eh, r**ai**n, **eigh**t
Θ, θ	Theta	Th, **th**eater, **th**under
I, ι	Iota	Ee, f**ee**t, j**ee**p
K, κ	Kappa	K
Λ, λ	Lambda	L
M, μ	Mu	M
N, ν	Nu	N
Ξ, ξ	Xi	Ks, du**cks**
O, o	Omicron	Oh, **oa**t
Π, π	Pi	P
P, ρ	Rho	R
Σ, σ	Sigma	S
T, τ	Tau	T
Υ, υ	Upsilon	U
Φ, ϕ	Phi	F
X, χ	Chi	Sh, **sh**opping
Ψ, ψ	Psi	Ps, cu**ps**
Ω, ω	Omega	O, b**o**ss

A.8 Special sets

Despite my informal notation, these are all sets

$$\mathbb{N} = \{0, 1, 2, 3, 4, 5, \ldots\}$$

$$\mathbb{Z} = \{\ldots, `5, `4, `3, `2, `1, 0, 1, 2, 3, 4, 5, \ldots\}$$

$\mathbb{R}$ any given number on the number line.

$$\mathbb{Q} = \left\{ \tfrac{x}{y} \in \mathbb{R};\ x, y \in \mathbb{Z} \wedge y \neq 0 \right\}$$

$$\mathbb{C} = \left\{ a + bi;\ (a, b) \in \mathbb{R} \times \mathbb{R}, i = \sqrt{`1} \right\}$$

$$\mathbb{I} = \mathbb{R} \setminus \mathbb{Q}$$

A.8.1 Properties and identities

Addition

Additive identity $a + 0 \equiv 0$

Associative property $a + (b + c) \equiv (a + b) + c$

Commutative property $a + b \equiv b + a$

Cancellative property $a + b = a + c \implies b = c$

Negative property $\forall a \in \mathbb{C};\ \exists `a \in \mathbb{C};\ a + `a = 0$

Distribution $`(a + b) \equiv `a + `b$

Subtraction

Definition $a - b := a + `b$

Associative property $a - (b - c) \equiv (a - b) - c$

Subtractive identity $a - 0 \equiv a$

Negative property $a - b \equiv a + `b$

Distribution theorem $a - (b + c) \equiv a - b - c$

Multiplication

Multiplicative identity $a \cdot 1 \equiv a$

Associative property $a \cdot (b \cdot c) \equiv (a \cdot b) \cdot c$

Commutative property $a \cdot b \equiv b \cdot a$

Cancellative property $a \cdot b = a \cdot c \implies b = c$

Divisive property $\forall a \in \mathbb{C};\ \exists \text{arc}\,(a) \in \mathbb{C};\ a \cdot \text{arc}\,(a) = 1$

$$\begin{aligned} \text{arc}\,(0) &:= 1 \\ \text{arc}\,(n) &:= \tfrac{1}{n} \end{aligned}$$

Distribution $\frac{a}{b \cdot c} \equiv \frac{a}{b} \cdot \frac{a}{c}$

Distribution over + $a \cdot (b + c) \equiv (a \cdot b) + (a \cdot c)$

Notation $ab = a \cdot b$ if a and b are two separate things.

Division

Divisive identity $a \div 1 \equiv a$

Separative property $a \div b \equiv a \cdot \frac{1}{b}$

Antiassociative property $a \div (b \div c) \equiv a \cdot (c \div b)$

Appendix B

Graph source code

This appendix contains all of the source code for the various graphs throughout the book.

```
#!/usr/bin/env sage

# the points
points = [(2,4), (-3,4), (-5,-1), (5,-3)]

# This next little bit constructs the labels for the points
labels = []                     # This is the list of labels.

# Loop through the points
for point in points:
    # The label needs to be slightly to the right of the point, as to
    # not overwrite it.
    newpoint = (point[0] + 0.1, point[1])
    # This label will just have the coordinate listed, with some
    # styling.
    this_label = text("$"+str(point)+"$", newpoint, fontsize=25,
                      rgbcolor=(0,0,0), horizontal_alignment="left")

    # Add this label to the list.
    labels.append(this_label)
labels = sum(labels)

# A plot of the points
myticks = [range(-100,100)] * 2
pts = list_plot(points,
                ticks=myticks,
                tick_formatter="latex",
                pointsize=25,
                axes_labels = ['$x$', '$y$']
               ) + labels
pts.set_axes_range(-6,6,-4,5)
pts.fontsize(25)
# Save it to a file
pts.save("VectorGraph2.png")
```

Listing B.1: This program puts a few points on a Cartesian coordinate plane.

```
#!/usr/bin/env sage

from numpy import arange

squarelist = []
textlist = []

for x in arange(-10,11):
    t = text('$' + str((x, x**2)) + '$',
             (x+0.3, x**2),
             rgbcolor='black',
             horizontal_alignment='left'
            )
    squarelist.append((x,x**2))
    textlist.append(t)

pts = list_plot(squarelist,
                ticks=2,
                tick_formatter=1,
                pointsize=15,
                color='red',
                axes_labels=['$x$', '$f(x)$']
               )
pts.set_axes_range(xmin=-11, xmax=11, ymin=-10, ymax=110)
pts.fontsize(15)
pts.save("x-squared-nolabels.png")

# Add labels
pts_with_labels = pts + sum(textlist)
pts_with_labels.save("x-squared-labels.png")

# Add connecting lines
pts_with_conn = pts_with_labels + list_plot(squarelist, plotjoined=True)
pts_with_conn.save("x-squared-joined.png")

# Add Curve
x=var('x')
pts_with_curve = pts + plot(x^2, (x, -11, 11), color='green')
pts_with_curve.save("x-squared-withcurve-nolabels.png")

curve = plot(x^2, (x, -11, 11),
             color='green',
             ticks=[list(arange(-20,20,2)), list(arange(0,140,20))],
             thickness=5,
             axes_labels=['$x$', '$y$'])
             # legend_label='$\\lambda (x) \\to x^2$')
curve.fontsize(25)
curve.save("x-squared-curve.png")
```

Listing B.2: Produces a number of graphs corresponding to $\lambda\,(x) \to x^2$

Appendix C

Basic arithmetic

This is a review appendix. It will teach you the basic facts of arithmetic, basic algebra, and how to do proofs. It's too boring for the rest of the book. Nonetheless, even if you have arithmetic and proofs under your belt, this chapter will be very helpful.

We are going to start with some very simple axioms about arithmetic, called the Peano axioms. From there, we will prove all of the things we know about addition, subtraction, multiplication, et cetera.

This is more or less a copy of Edmund Landau's *Foundations of Analysis*, found in [14]. However, Landau's book, while very rigorous, is very breve, and very dry. His book is about 130 pages long, and very formal. This appendix is unfinished; however, when it is finished, I expect it to be much longer, and very informal — but nonetheless rigorous.

Even if you don't read this, I highly recommend you buy a copy of Landau's book, if only for reference purposes. It doesn't cost very much. I think I bought my copy for US $30.00.

This appendix is independent of the rest of the book – the main part of the book does not assume you have read this appendix, and this appendix doesn't assume you've read the rest of the book.[1] (Hence why it's an appendix). With that in mind, there is some duplication between here and the book. Sorry about that.

[1] Although the book does assume you know most of the stuff covered in this appendix.

C.1 Peano axioms

Properties of equality

Before we get to the slightly less boring part, we have to review the properties of equality.

$x = y$ means that two things — x and y in this case — are the same thing, at least in some scope.

If I use a letter instead of a number, it usually means "stick some number here, but we don't know what number it is". If it's in the context of "for all", then it usually doesn't matter what number we are talking about, as the property is true for every case.

Reflexive property $x = x$, for all x. So, x is the same thing as itself. Duh.

Commutative property For all x and y, if $x = y$, then $y = x$. "Commute" means "move", so the commutative property is the property of moving things around.

Transitive property For all x, y, and z, if $x = y$, and $y = z$, then $x = z$.

Thus, something like

$$a = b = c = d$$

Is just the lazyman's way of writing

$$a = b, b = c, c = d$$

Because of the transitive property, it also means

$$a = c \wedge d = b \wedge a = d$$

Axioms of natural numbers

The natural numbers are the "whole numbers", usually denoted as $\mathbb{N}$.

$$\mathbb{N} := \{0, 1, 2, 3, 4, \dots\}$$

Axiom 1. 0 *is a natural number.*[2]

Axiom 2. *For each natural number x, there is exactly one separate natural number, called the successor of x, denoted* $\mathcal{S}(x)$.

The successor is the next number. So, $\mathcal{S}(0) = 1$, $\mathcal{S}(1) = 2$, $\mathcal{S}(2) = 3$, *et cetera. This also means that we can define every natural number as some succession from* 0*:*

It's also true that if $x = y$, *then* $\mathcal{S}(x) = \mathcal{S}(y)$. *(This makes* $\mathcal{S}$ *a function).*

$$\mathcal{S}(\mathcal{S}(\mathcal{S}(\mathcal{S}(0)))) = 4$$

Axiom 3. *There are no two numbers who have the same successor. That is, if* $\mathcal{S}(x) = \mathcal{S}(y)$, *then* $x = y$, *for all x and y. (This makes* $\mathcal{S}$ *an injection).*

Axiom 4. *There is no natural number q such that* $\mathcal{S}(q) = 0$.

Axiom 5. *Let there be a set M such that:*

1. 0 *is in M (denoted* $0 \in M$*)*
2. *If some number x is in M, then its successor* $\mathcal{S}(x)$ *is also in M*

Then M contains all of the natural numbers. This establishes the completeness of $\mathbb{N}$.

[2]Some people say that 1 is the first natural number. It doesn't matter a whole lot, at least as far as construction goes. Most people nowadays start with 0, because 0 is the additive identity. That is, $a + 0 \equiv a$.

C.1.1 Addition

Theorem 1. *For all x and y, if* $x \neq y$*, then* $\mathcal{S}(x) \neq \mathcal{S}(y)$.

Proof. Else we would have $\mathcal{S}(x) = \mathcal{S}(y)$, and, by axiom 3, $x = y$ □

Theorem 2. $\mathcal{S}(x) \neq x$

Proof. Let Q be the set of all x for which this property holds true.

By axiom 1, $0 \in \mathbb{N}$. By axiom 3, $\nexists\, q \in N;\ \mathcal{S}(q) = 0$. Therefore $\mathcal{S}(0) \neq 0$.

By construction, if $x \in Q$, then $\mathcal{S}(x) \neq x$. By the previous theorem, $\mathcal{S}(\mathcal{S}(x)) \neq \mathcal{S}(x)$, which would mean that $\mathcal{S}(x) \in Q$. Thus, by axiom 5, $Q = \mathbb{N}$.

Therefore, for all $x \in \mathbb{N}$, $x \neq \mathcal{S}(x)$ □

This is unfinished.

Appendix D

Answers to the exercises

Answer (Ex. 1) — Start with the first law

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$

Let $p = \neg a$, $q = \neg b$. One of the rules of algebra is, if $x = y$, then you can substitute one in for the other.

$$p \vee q \equiv \neg(\neg p \wedge \neg q)$$

Another rule is, if you have an equation, and you do something to one side of the =, you have to do the same thing to the other side. Here, we're going to apply $\neg$ to both sides of $\equiv$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

Answer (Ex. 2) — Start with the first property

$$a \wedge (b \vee c) \equiv (a \wedge b) \vee (b \wedge c)$$

Apply $\neg$ to both sides

$$\neg(a \wedge (b \vee c)) \equiv \neg((a \wedge b) \vee (b \wedge c))$$

Apply DeMorgan's laws

$$\neg a \vee \neg (b \vee c) \equiv \neg (a \wedge b) \wedge \neg (b \wedge c)$$

Do it again (inside the parentheses).

$$\neg a \vee (\neg b \wedge \neg c) \equiv (\neg a \vee \neg b) \wedge (\neg b \vee \neg c)$$

Let $p, q, r = \neg a, \neg b, \neg c$, respectively.

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (q \vee r)$$

Answer (Ex. 3) — Yes.

*Proof.*By transition,

$$\begin{array}{lllll} \neg A & \implies & B & \implies & \neg C \\ \neg A & & & \implies & \neg C \end{array}$$

By the reversal property:

$$A \impliedby C$$

□

Answer (Ex. 4) — Not by necessity.

*Proof.*By transition:

$$\begin{array}{lllll} A & \not\implies & B & \implies & \neg C \\ A & \not\implies & & & \neg C \end{array}$$

This does not imply

$$A \implies C$$

To put it another way

$$\begin{array}{ccc} A & \not\implies & \neg C \\ & \not\Downarrow & \\ A & \implies & C \end{array}$$

□

Answer (Ex. 5) — No, there's nothing to indicate that A and C have any relation whatsoever.

Answer (Ex. 6) — No

*Proof.*Let's assume that it's true.

$$A \wedge \neg [B \wedge (C \vee D)] \quad \equiv \quad A \wedge (B \vee C) \wedge (B \vee D)$$

Let B, C, D all be true. Then:

$$\begin{aligned}
A \wedge \neg [\text{True} \wedge (\text{True} \vee \text{True})] &\equiv A \wedge (\text{True} \vee \text{True}) \wedge (\text{True} \vee \text{True}) \\
A \wedge \neg [\text{True} \wedge (\text{True} \vee \text{True})] &\equiv A \wedge \text{True} \wedge \text{True} \\
A \wedge \neg [\text{True} \wedge (\text{True} \vee \text{True})] &\equiv A \wedge \text{True} \\
A \wedge \neg [\text{True} \wedge \text{True}] &\equiv A \wedge \text{True} \\
A \wedge \neg \text{True} &\equiv A \wedge \text{True} \\
A \wedge \neg \text{True} &\equiv A \wedge \text{True} \\
\text{False} &\equiv \text{True}
\end{aligned}$$

Which is obviously false. □

Answer (Ex. 7) — Let's look at $f : A \to B$. If f is surjective, then $B = \text{im}(f)$, so we can write

$$f : A \to \text{im}(f)$$

In other words

$$f : \textbf{dom}(f) \to \text{im}(f)$$

It must be true that f is a surjection for f to be invertible. Else there would be elements in the codomain of f that were not in the domain of $\text{arc}(f)$.
We've established

$$\text{arc}(f) : \text{im}(f) \to \textbf{dom}(f)$$

Let's assume f is invertible. Then $\textbf{dom}(f) = \text{preim}(f)$. Thus

$$\text{arc}(f) : \text{im}(f) \to \textbf{dom}(f)$$

For $\operatorname{arc}(f)$ to be a function — i.e. for f to be invertible, then it must be true that

$$\nexists\, a,b \in \operatorname{im}(f);\ a = b;\ \operatorname{arc}(f,a) \neq \operatorname{arc}(f,b)$$

If we flip this around

$$\nexists\, a,b \in \operatorname{preim}(f);\ a \neq b;\ f(a) = f(b)$$

That is, the definition of injectivity. Thus we have proven

$$f \text{ is invertible} \iff (f \text{ is an injection}) \wedge (f \text{ is a surjection}) \iff f \text{ is a bijection}$$

Bibliography

[1] Aleph null. URL: https://en.wikipedia.org/wiki/Aleph-null (visited on 03/05/2015).

[2] Alpha conversion. URL: https://wiki.haskell.org/Alpha_conversion (visited on 03/01/2015).

[3] Beta reduction. URL: https://wiki.haskell.org/Beta_reduction (visited on 03/01/2015).

[4] Boolean algebra. URL: https://en.wikipedia.org/wiki/Boolean_algebra (visited on 03/01/2015).

[5] Currying. URL: https://en.wikipedia.org/wiki/Currying (visited on 02/23/2015).

[6] Eta conversion. URL: https://wiki.haskell.org/Eta_conversion (visited on 02/23/2015).

[7] Eta conversion. URL: https://wiki.haskell.org/Eta_conversion (visited on 03/01/2015).

[8] Galileo's Paradox. URL: https://en.wikipedia.org/wiki/Galileo%27s_paradox (visited on 03/08/2015).

[9] Greek alphabet. URL: https://en.wikipedia.org/wiki/Greek_alphabet (visited on 03/01/2015).

[10] Greek government-debt crisis. URL: https://en.wikipedia.org/wiki/Greek_financial_crisis (visited on 02/23/2015).

[11] Thomas Jech. Set Theory. New York, NY: Springer, 2003. ISBN: 3-540-44085-2.

[12] Lambda abstaction. URL: https://wiki.haskell.org/Lambda_abstraction (visited on 03/01/2015).

[13] Lambda calculus. URL: https://wiki.haskell.org/Lambda_calculus (visited on 03/01/2015).

[14] Edmund Landau. Foundations of Analysis. Providence, RI: AMS Chelsea Publishing, 1966.

[15] Miran Lipovača. Learn You a Haskell for Great Good! San Francisco, CA: No Starch Press, 2011.

[16] Operator associativity. URL: https://en.wikipedia.org/wiki/Operatorassociativity (visited on 02/23/2015).

[17] David Pengelley Reinhard Laubenbacher. Mathematical Expeditions: Chronicles by the Explo New York, NY: Springer, 2000. ISBN: 0-387-98433-9.

[18] steve jobs on programming. URL: https://www.youtube.com/watch?v=5Z1gfgM7kzo (visited on 01/01/2015).

[19] Zermelo-Fraenkel set theory. URL: https://en.wikipedia.org/wiki/Zermelo%E2%80%93Fraenkel_set_theory (visited on 02/23/2015).