Intro to Proofs for the Morbidly Curious

Evan Chen《陳誼廷》

19 January 2024

> It's weird how we taught students to encode their understanding of math in their understanding of social permission (what they're "allowed" to do). Permission has nothing to do with it. The question is what procedures turns true statements into true statements.
>
> Qiaochu Yuan, on Twitter

Contents

§1 The tl;dr

For your purposes, a **mathematical proof** is an *informal* but *convincing* natural-language explanation for why a statement is true.

If you are learning how to write proofs, I want to emphasize **you could stop reading here** and, in the hands of a competent mentor, you would probably be okay. But I realize the sentence above is evasive: what exactly does "convincing" mean?

So I'll tell you exactly what "convincing" means, in full gory detail that could leave you sorry that you asked. However, I want to emphasize that **this verbosity is mostly to satisfy your curiosity**. It shouldn't change how you approach mathematics.

§1.1 Roadmap

The thesis of this article is captured in the following "definition":

Definition 1.1. A **mathematical proof** is an **informal natural-language explanation** that *in principle* could be **compiled** into a **formal machine-verifiable proof**.

In Section 2 we provide some analogies that try to give explain what "compiled" means in this big picture. Then in Section 3 we describe how "informal natural-language explanation" is done in practice. Finally, in Section 4 we talk a bit about what "formal machine-verifiable proof" means, just as a curiosity. At the end, Section 5 gives some pragmatic advice for actually learning how to write proofs.

§2 Some analogies about abstraction and compilers

To proceed further, I need to talk about abstraction — the idea that humans can work "several levels higher" than the "base" of objects they are manipulating. So here are two imperfect analogies.

§2.1 Analogy from programming

Suppose you're writing a Hello World program in C++, like

```
1 #include <iostream>
2 int main() { std::cout << "Hello World!"; return 0; }
```

What happens after you write this code? Well, after that, the compiler will translate the program into *machine code*. And this machine code is incomprehensible to humans. For example, an intermediate step in the compilation process might start with:

```
  .file  "hello.cpp"
  .text
  .local  _ZStL8__ioinit
  .comm  _ZStL8__ioinit,1,1
  .section  .rodata
.LC0:
  .string  "Hello World!"
  .text
  .globl  main
  .type  main, @function
main:
.LFB1761:
  .cfi_startproc
  pushq  %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset 6, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register 6
  leaq  .LC0(%rip), %rax
  movq  %rax, %rsi
  leaq  _ZSt4cout(%rip), %rax
  movq  %rax, %rdi
  call  _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc@PLT
```

Holy cow! And that's just one of the intermediate steps. The final output is going to be some bytes run by machine.

It's good that programmers are told this is vaguely what's happening underneath the hood, but for practical day-to-day work, they don't need to actually understand how the machine code works. Programmers never read the generated machine code — and they don't need to. **Instead, these programmers "think" in C++, and then simply trust the compiler will do the work of interpreting the machine code.** These programmers neither know nor care how that compiler works. That tough work is deferred to people who write compilers.

Proofs in math operate on a similar principle, except the compilers are imaginary humans that understand English. That is: you provide a natural-language explanation of why a statement is true. This explanation should, *in principle*, be able to compiled into a "formal machine-verifiable proof" – the analog of machine code. We describe in Section 4 what this is (spoiler: it's not fun to read). But for a working student, the exact details of the compilation process are unimportant; it only matters that it *could* be done in theory.

§2.2 Analogy from Western chess

Another metaphor you can think about is Western chess. Officially, a game of Western chess "is" a sequence of moves in algebraic notation, such as:

```
1.  e4   e5
2. Nf3  Nc6
3. Bb5   a6
```

and so on, ending once one of the kings is checkmated (or draw, resignation, etc.). And there is a list of rules that govern what moves are legal — rooks may move only along a row/column, bishops along diagonals, and so on. The list of legal moves is not up for debate, it's just a social convention.

However, when people talk about strategy for chess, they don't speak primarily in algebraic notation, or spend a lot of time thinking about whether moves are legal. Coaches for chess speak in much broader strokes, saying things like "try to control the center" and "develop pieces", and the chess community gives names to common openings like "Morphy defence" to capture entire common sequences of moves into a single unit. In other words, people can naturally operate at a layer of abstraction above individual moves.

The advantage of this analogy over the programming analogy is that it gives a better idea of how vague notions like "control the center" still make sense for humans. The disadvantage of the analogy is that algebraic notation does turn out to be pretty understandable by humans, whereas formal proofs (like machine code) are incomprehensible.

§3 How natural-language proofs are done in practice

§3.1 The importance of definitions

Before starting on proofs, arguably it's more important to talk about *definitions*, especially since a lot of confusion comes from not having solid definitions.

An example of a definition in geometry might look like:

Definition 3.1. The *orthocenter* of a triangle is the intersection point of the three altitudes of the triangle.

But of course you can then ask further:

- What's the definition of an "altitude"? We define the altitude to be the line through a vertex perpendicular to the opposite side.
- What's the definition of "perpendicular"? The two lines should meet at a 90° angle.
- What's the definition of "angle"? Well...

At some point, you have to stop somewhere. And there is a way to stop somewhere, which we'll write about more in Section 4.

However, **for day-to-day work, these foundations are out-of-scope**. Instead, what people usually do is accept *some* concepts as "obvious", and don't give a definition for them, instead just trusting intuition. So, for example, probably no one ever gave a perfect definition of an "angle", even though we're happy to say that a 10° angle and 40° angle should add up to a 50° angle.

What counts as "obvious" for you is essentially a social convention, because the "compiler" is an imaginary human and not a real one. If you're in high school, for example, the statement $2+2=4$ is OK to take for granted, even though probably no one ever defined what plus, equals, 2, or 4 meant for you.

As a less obvious example, consider the statement

$$0.99999\ldots = 1.$$

This statement is true, and it comes as a surprise to a lot of people. But the reason they're confused is because nobody ever told them a proper definition of a repeating decimal (or "real number", for that matter). In fact, the definition of $0.99999\ldots$ means that it's equal to the infinite sum

$$\frac{9}{10} + \frac{9}{10^2} + \frac{9}{10^3} + \cdots$$

and the definition of an infinite sum is "the limit of the partial sums" (and the definition of "limit" is the one given in any analysis textbook). If you agree with all the definitions, you can check that limit indeed equals 1.

That shows that definitions themselves are really a social convention, too. In mathematics, once the definitions are chosen, the conclusions that follow from it are no longer up for debate. If you want to insist that $0.99999\ldots \neq 1$, then trying to debate the *proof* of this result is futile. The proof is correct. Instead, you need to propose a different definition of the left-hand side and then argue that your definitions are (subjectively) better than the standard definitions.

§3.2 Proving things after you have a few things taken for granted

As we said, in practice what we *actually* do is accept some objects as "undefined but obvious", and then build new definitions from there.

Let's give an example. In a number theory classroom, the instructor decrees that the set of integers $\mathbb{Z}$, and the operations $+$, $-$, $\times$, will be taken for granted, and are not up for debate. If you accept this, then you could make new definitions like:

Definition 3.2. An integer n is *even* if $n = 2 \cdot k$ for some integer k.

From this, you can deduce basic facts like:

Fact 3.3. The product of two even integers is even.

Proof. Because $2a \cdot 2b = 2 \cdot (2ab)$. □

Fact 3.4. The number 0 is even.

Proof. Notice that $0 = 2 \cdot 0$. □

Again, enemies of the concept that 0 is an even number (and there are many) will make no progress arguing about the *proof*, and instead need to propose a new definition of "even" and argue why their definition (subjectively) makes more sense.

§3.3 Proving things in real life

That's fine and all, but a lot of the readers of this article are probably proving results a lot more complicated than "product of two even numbers is even".

For example, here is an example of a competition problem:

Problem 3.5 (European Girls MO 2020)**.** The positive integers a_0, a_1, a_2, …, a_{3030} satisfy

$$2a_{n+2} = a_{n+1} + 4a_n \text{ for } n = 0, 1, 2, \ldots, 3028.$$

Prove that at least one of the numbers a_0, a_1, a_2, …, a_{3030} is divisible by 2^{2020}.

This is a lot more involved than the really simple example we just gave. If you had to include a justification the product of two even numbers is even, and then that the sum of two even numbers is even, and so on, you would end up with a short novella. It's just not feasible in practice.

So, what happens is again more social conventions. The student at the competition, and the grader for that student, both agree that there is no need to write out the proof that the sum of two even numbers is even: everyone knows it's true, and the student *could* write it out if for some reason they had to, but is omitting it for the sanity of everyone involved.[1]

So how much detail should a student need to show? The answer is, again, it's a social convention: it depends on your audience. For math competitions, the standard I use if I'm the grader is: "is there a chance that the student actually doesn't know how to prove this and is just bluffing to me?". If I think they could plausibly be bluffing, then I deduct; otherwise, I don't mind.

§4 How formal proofs are done in theory but not in real life

So what happens at the foundations? For example, earlier I mentioned that the statement $2+2=4$ is not subject to proof in a normal high-school classroom, because the definitions of 2, 4, +, = were never given. I'm sure that leaves you wondering just what those definitions are.

As we mentioned, the problem you face when you drill all the way down is that definitions usually depend on earlier definitions, and the buck has to stop somewhere. So what's the plan?

[1]For concreteness, the solution to the problem that I had written looks like this:

Replace 1010 by N in the obvious way and proceed by induction on $N \geq 0$ with the base case $N = 0$ being vacuous. Notice that for any index $k \neq 0, 3N-1, 3N$ we have

$$a_k = 2a_{k+1} - 4a_{k-1} = 2(2a_{k+2} - 4a_k) - 4a_{k-1} = 4(a_{k+2} - 2a_k - a_{k-1})$$

so it follows that $(a_1, a_2, \ldots, a_{3N-2})$ are all divisible by 4 and satisfy the same relations. But then $(a_1/4, a_2/4, \ldots, a_{3N-2}/4)$ has length $3(N-1)+1$ and so one of them is divisible by 4^{N-1}; hence some term of our original sequence is divisible by 4^N.

§4.1 Solution: specify axioms you want to be true

For concreteness, let's specialize to an example: set theory. Leinster's textbook *Basic Category Theory* has a great exposition of this in Chapter $2\frac{1}{2}$, that I abridge here.

Often, the concept of a "set" is taken to be a primitive concept: we *think of* a set as "an unordered collection of objects", but we do *not* give an official definition of a set. Instead, what we do is specify a *wishlist* of properties that we want sets to have, based on our intuitive understanding. We consider these as axioms, and the most popular set of axioms in use today is called *ZFC*.

The way ZFC works is that we specify a new relation "$\in$", which means membership, and use that to state our axioms. Here are a few examples of axioms that are in ZFC:

- **Empty Set Axiom**: There exists a set $\varnothing$, such that for any x, the statement $x \in \varnothing$ is false.
- **Axiom of Extension**: If A and B are sets, and $x \in A$ if and only if $x \in B$, then $A = B$.

And so on. The list is long and complex (you can find it on the Internet if you want to), but the point I want to drive home is that **rather than define "set" and $\in$, we agree on a wishlist of axioms**. Like with definitions, these are social conventions, not subject to "truth". However, once the axioms are set in stone, then you can start proving things using those axioms, and build the rest of mathematics out of it.

§4.2 First-order logic

We've now seen that logicians can specify a primitive concept like "$\in$", not based on defining what $x \in A$ means, but instead by agreeing on a wishlist of properties that $\in$ should have, and making deductions from there.

To then go even further, logicians also encode sentences using symbols.

- $\forall$, the symbol "for all", or $\exists$, "exists";
- logical symbols like $\iff$ (for "if and only if"), $\neg$ (for "not"), parentheses, and so on;
- an equality symbol $=$;
- variable names,
- ...

That means, for example, the "empty set" axiom earlier could be written as the sentence

$$\exists A \forall x \neg (x \in A).$$

Mathematical machine code! And like before, logicians then specify a "wishlist" of properties they want these sentences to satisfy. Again, they don't *define* what the symbols mean; instead, the specify some rules about how sentences can be transformed. For example, the "reflexive property" $a = a$ is taken as an axiom, as is the "symmetric property" $(a = b) \iff (b = a)$, and so on.

This lets you give a truly airtight definition of proof, in the sense that it could be verified by a computer:

Definition 4.1. A **formal proof** is a sequence of sentences, each of which is either an axiom or follows from preceding sentences by a specified transformation rule.

§4.3 Real life

The earlier section is so far removed from reality that working mathematicians, or students in mathematics, will never need to think much about it.

But the punch line is that a proof is something we think *should* be compilable to a formal proof in principle, even if we never do this compilation in practice.[2]

Here, the compiler is an imaginary human who's an expert in formal logic, and has an infinite amount of time and patience. They should be able to take the writing you gave them and, slowly but surely, compile it into an incomprehensible mess of symbols like above, without having to come up with additional ideas.

§5 Miscellaneous advice for how to learn to write proofs

Here's a few words of advice on how to actually learn how to write proofs. If you'd like a book to follow, I like Rotman's *Journey into Mathematics*[3].

Here's a few other notes:

§5.1 What should I do?

Read the proofs to problems you think you've solved as examples to learn from. Note that these don't have to be from proof contests! The official solutions to any respectable contest would all pass as proofs. Then, if you can, write your own proofs and try to find a mentor who can check them and provide feedback.[4] (This is admittedly pretty hard to do, because good mentors are hard to find.)

§5.2 What makes a proof "rigorous"?

Honestly, if you're new to proofs, it's probably better for your learning if you just replace the word "rigorous" with "correct" everywhere.

One big mistake I think you can make in the learning process is to confuse yourself into thinking that you can have a correct proof, but somehow need to add some magic words to make it "rigorous". That's not how it works. The goalpost is that your explanation should compile to machine code. If you meet that goal, you are both correct and rigorous. If you fail that goal, you don't have a proof.

§5.3 How do I know when a step is "legal"?

This is a question I see sometimes, because I think students and teachers alike would love it if there was some simple deterministic rule you could use to check a step. Unfortunately, formal machine-verifiable proofs are so far removed from how humans actually work, it's not practically feasible to do this using some fixed procedure.

Nonetheless, I do have a couple of common cases where I can offer a bit of advice:

[2]Actually, this is a white lie. There *is* some effort to make machine-verification of proofs a reality; systems like Coq and Lean were built to try to make this process at least plausible in some areas of math. And they're making steady progress!

[3]Available at https://store.doverpublications.com/0486453065.html.

[4]Also, I better make a comment on style. Since you're writing in natural language, there are cosmetic ways you can make your proof easier to read, such as being clear with definitions, adding paragraph breaks, and isolating clear claims, and so on.

Good mentors will make notes of this to help you, and you should listen, but I also want to say that you should try to keep a distinction in your head between stylistic comments on how to be clearer about what you mean, versus real errors in logic and math.

- First, **check if you understand the definitions of what you're using**. As an example, a student might ask whether you are "allowed" to take the statement

$$0.333\ldots = \frac{1}{3}$$

and multiply both sides by 3 to prove that $0.999\ldots = 1$. The answer is "yes", but if you need to ask this, then chances are you don't actually know the definition of a real number or infinite sum[5], and that's your real issue (no pun intended). You won't be able to understand why the answer is "yes" until you fix this.

Depending on where you are in your mathematical education, sometimes you will just have to take your teacher's word for it, because e.g. a certain definition could require a one-semester university course in analysis or measure theory.

- If you understand all the definitions, **try to prove the step is legal**. If you can't, it's likely that something is wrong.

For example, one of the proof keywords you'll run into sometimes is the phrase "without loss of generality" (often abbreviated WLOG). What this phrase really means is: "these cases are obviously exactly the same, so we only do one of them". For instance, sometimes you need to prove a statement about two real numbers a and b that's symmetric, so you can say "WLOG $a \geq b$"; because in the case where $b \geq a$, all you do is switch the letters a and b everywhere in your proof.

That means, if you have *any doubt at all* about whether a WLOG is legal, then something is wrong.

- Also, **check whether the step made is actually a true statement**! This might sound obvious, but you can get a surprising amount of mileage by spending a minute or two seeing if you can find a counterexample. If you discover a counterexample, that gives you immediate confirmation that there's a mistake.

As a grader for proofs written by students, I do this all the time. Often times I read a sentence by the student, and I can't see why it follows from the student's previous work. So then I have to figure out whether I've overlooked something (it happens), or whether the student made a mistake. Usually, the first thing I try to do is *not* work out the logic; instead, I try to conjure a counterexample. This works more often than you'd expect, and when it does, I can be totally confident it's the student's mistake and not mine.

§5.4 Is there a way to verify entire proofs without having to check every individual step?

Again, there will be no surefire way to do this rapidly. However, like in the last section, there are some things you can use *in practice* to quickly sanity-check your work. Passing all the pre-tests isn't a guarantee there's no mistake, but these pre-tests seem to be pretty robust in practice and can give you a bit more well-placed confidence.

(While I was writing this, I realized that a lot of these actually appear in Scott Aaronson's blog post Ten Signs a Claimed Mathematical Breakthrough is Wrong. That blog post is about verifying research papers, but some of the advice still applies.)

- **Look for counterexamples to statements you doubt** (we said this already). I can't emphasize this enough.

[5]General heuristic: if you're doing anything related to probability or infinite sums, and you don't know any university-level math, then you probably don't understand the definitions.

- In fact **look for nearby statements which are false too**. In philosophy, we call this "proving too much". In other words, you should ask yourself the question: "*if this approach did work, would it also imply a similar result that's false?*".

  For example, in a recent episode of Twitch Solves ISL, there was a set A of positive integers which we needed to prove had a property $\mathcal{P}$. We had defined a certain notion of "density", and we knew that $d(A) \approx \frac{1}{13}$. So one audience member tried to suggest an argument using the fact that $d(A) < \frac{1}{10}$ to deduce $\mathcal{P}$. We immediately replied by giving a counterexample of a set B not satisfying $\mathcal{P}$ where $d(B) = \frac{1}{11}$.

  Even though B was unrelated to the problem at hand, it showed that $d(A) < \frac{1}{10}$ could not *by itself* solve the problem. A grader who sees a student trying to only use $d(A) < \frac{1}{10}$ already knows the proof is wrong without reading it.

- For longer and more difficult problems, try to **isolate key claims and their proofs**, when possible. If you look at any solutions published by me, you will often see that there are big claims marked visibly in green boxes, and self-contained proofs of these claims.

  Why is this useful for checking your work? There are two reasons.

  - First, when you identify *what* the key claims are, you can hold them up to scrutiny by, e.g. searching for counterexamples to them.
  - Secondly, small components are easier to check than their union. Verifying a proof is exponentially long in the number of moving parts, so any time you can "divide and conquer", it's almost always good.

- For competition problems, **identifying critical steps** is especially important because those problems tend to have one or two key ideas.

  As an example, I was a grader for the recent geometry problem USAMO 2023/1. We knew that the "standard" approach involved chasing some angles, then doing a big key step of combining equating certain lengths or ratios, and concluding from there. The lengths and ratios looked different from one approach to another, but every correct solution seemed to have a crux move related to lengths.

  That meant that, if a student tried to solve the problem using *only* angles without involving any lengths or ratios, I knew right away something was amiss. *Angles alone shouldn't be able to solve this problem*, I told myself. If you want to get from Boston to Beijing, you can't just drive a car. There's a point you need to an airplane (or at least a boat) in order to cross the ocean.

  The way you find the mistake in such proofs is to look for the big gap. Classify each statement as "within driving distance of Boston" versus "within driving distance of Beijing", and look for the step where we suddenly switched between them.

- **Write your solutions out, do not just keep them in your head.** I don't trust myself to have solved a problem completely anymore until I have the solution in writing. I cannot tell you how many times I only found a mistake once I started actually typing up what I thought was a complete solution.

§5.5 What about methods of proof?

Oh yeah — proof by induction, proof by contradiction, all that, right?

Yeah, you should learn all of them, and you'll run into them all eventually. But I want to be clear: **proof strategies are not new rules**. They are *strategies*.

It's like learning the "Sicilian defense" in chess. You're playing the same game of chess, the rules of chess didn't change. The Sicilian defense is just a pattern that you can use which is so common that it earned its own name.

§5.6 What subjects are good to learn proofs with?

Graph theory is really good. Elementary number theory is also a nice choice. Meanwhile, real analysis and set theory are among the worst possible choices.

§5.7 Any recommendations on how to write more clearly?

Yes, that's in a different handout:

https://web.evanchen.cc/handouts/english/english.pdf

§5.8 Any last words of advice?

Don't be scared! Many of the students I've work with simply self-taught themselves proofs by reading enough examples and using logical common sense.

It's really not supposed to be difficult, just like learning the rules of chess is not supposed to be difficult (whereas learning to play chess *well* can take a lifetime). As I promised at the start of the article, you could have stopped reading after the first sentence and been fine.