Proof-Writing Guidelines

1 Introduction

A (mathematical) proof of a statement is simply an argument, presented as an essay, that convinces the reader of the validity of the statement. Therefore, whether a proof is acceptable or not (e.g. whether it has the right level of detail, whether it is written in good style, etc.) is a subjective matter without very strict rules.[1] This makes the task of writing good proofs (i.e. proofs that are correct and easily understood and verified) a difficult one.

As you will notice when you read this document, there are a lot of similarities between writing a correct computer program with good style and writing a correct mathematical proof with good style. Unfortunately, however, writing a good proof is arguably harder since there is no compiler to point out all the compile-time errors you have (e.g. syntax errors or type-checking errors). Furthermore, you cannot run a mathematical proof on a computer to see if it produces the expected output. All the errors in a mathematical proof must be caught by you, the author of the proof, and this requires an extra level of attention to detail.

Our goal in this document is to give you suggestions on how to write correct and clearly presented proofs, and help you more easily catch logical flaws or gaps in your arguments. Most of our suggestions are stylistic in nature, because a proof that is written using good style is a proof that exposes its bugs (if there are any). So by following our guidelines, you will make it easier for yourself and everyone else to understand and validate your proof.

Mathematics, sciences and arts all have deep, complicated and beautiful ideas. Our rate of progress as a species depends on clear communication of those ideas so they can

[1] A remark is in order. It is widely accepted that mathematical proofs can be completely formalized in a way that can be mechanically verified, e.g. by a computer. However, writing proofs in a completely formal way is like writing computer programs using machine language. Mathematicians do not communicate proofs in that level of detail.

quickly spread and evolve. We hope that you will seek clarity of thought and effective communication of ideas, not just in the proofs you write, but in all of your endeavors in life.

2 Guideline Points

This section contains the guideline points (10 of them) that we ask you to follow when you write your proofs. Before we start, below is an example of a "proof" that violates almost all the guideline points. Feel free to refer back to this example as you go through this section.

Exercise ($2^n > n$)**.** Show that $2^n > n$ for all integers $n \geq 1$.

Proposed solution. (Line numbers are added for easy referencing.)

1. $F_n =$ "$2^n > n$"
2. $F_1 =$ "$2 > 1$" ✓
3. $F_n \implies F_{n+1}$:
4. $2^{n+1} = 2 \cdot 2^n > 2 \cdot n$ (induction) $\geq n + 1$ because $n \geq 1$
5. Therefore proved.

Keep in mind that the above example is really a toy example.[2] Some of the points we make below have more meaning and value in the context of more sophisticated proofs, which are the kinds of proofs you will be writing in CS251.

2.1 Is the basic structure right?

A proof is not a calculation or a sequence of mathematical symbols. A proof is an essay! It is an argument written in some human language (which, in this course, is English). This means that your proof should consist of paragraphs,[3] and every paragraph should consist of full English sentences. Every word and mathematical notation must be part of a complete sentence. The only purpose of mathematical notation/symbols is to make your arguments clear and concise. Do not equate mathematical notation with rigor or formalism. You can write a completely rigorous and formal proof using just English words.

Pictures and/or diagrams are highly encouraged when they help clarify your argument. However, a picture or a diagram does not replace an actual argument that needs to be clearly specified in English.

2.2 Are you starting the right way?

Often, the most important step in coming up with a proof is making sure that you understand exactly what assumptions are given to you and what statement needs to be derived. To make these things absolutely clear to you (the author of the proof) and the reader, we expect the first paragraph of your proof to include a restatement of the assumptions given and what needs to be derived. Unpack the definitions of technical terms if appropriate.

Another important component of the first paragraph is stating your general proof strategy (e.g. proof by contradiction, proof by induction, etc.) For proofs by contradiction, explicitly negate the statement that you are trying to prove and assume it. For proofs by contrapositive, the contrapositive of the statement should be explicitly laid out. For proofs by induction, the parameter being inducted on should be clear.

[2] However, this example is in fact not made up. It was a turned-in solution in CS251 many years ago.
[3] Avoid using long paragraphs as they hurt the clarity of the exposition.

2.3 Did you read your proof out loud?

When someone reads your proof, they will read it like they read any other essay. Therefore your proof should have a good flow and should be easy to read out loud even with the mathematical notation interspersed in the sentences. In particular, the mathematical notation you use or the diagrams you draw should not break the flow.

2.4 Is the purpose of every sentence clear?

The purpose of every sentence should be clear *as you are reading it*. Otherwise, the sentence can be very confusing for the reader and break the flow of the proof. A common example that violates this point is to make a statement without clarifying whether it follows from the previous statements or assumptions, or whether it is a statement that will be proved later on. With this in mind, whenever you write a sentence, make sure that it is clear if the sentence is (i) an assumption, (ii) a statement that follows from or combines previously established statements or assumptions, (iii) a claim that will be proved later, (iv) a sentence setting up a goal, (v) a sentence introducing a new variable, terminology, definition etc. (vi) part of an example or illustration, (vii) a digression, or (viii) something else.

A strategy that helps a lot is to set explicit and clear goals, and say what you will do *before* doing it.

2.5 Are you giving the right level of detail?

What is the right level of detail to provide when writing a proof? This is an important question whose answer depends on the audience of the proof. It is possible to write a proof that has all the right ingredients, but does not have the proper justifications for each step. So then the reader has to check the details themselves and verify that the proof is indeed correct. These kinds of proofs are actually not uncommon in, say, computer science or mathematics publications. The author may knowingly give just enough detail in a proof so that the gaps can be figured out and verified by an expert in the field. In CS251, however, we do not accept such proofs. No extra effort/verification from the reader should be necessary. For this reason, it is better to err on the side of caution, and spell things out as much as you feel is reasonable.

To help with this, our recommendation is that you view your audience as a classmate who does not know how to prove the statement you are presenting the proof for. Keep your audience in mind when presenting your proofs and provide the appropriate level of detail in your arguments. Your classmate should be able to read your proof from start to finish *once*, and be convinced that your argument is correct.

One thing that can be hard to appreciate is that when you think hard on a problem, you develop a lot of intuition in the process, and certain things that were not too trivial in the beginning start becoming obvious to you. This is great, because it signals that you are acquiring a deeper understanding of the problem. However, when it is time to write down your proof, you have to keep in mind that the reader has not gone through the mental process that you have gone through trying to come up with the proof. So the intuitions you have built are not necessarily accessible to your reader. And the things that are obvious to you may not be obvious to the reader. It is not always easy, but try to put yourself in the shoes of your reader and don't skip over details that can be crucial to understanding your argument.[4]

Related to the point above, if in your proof you are using a word that is synonymous to "obvious", double check (with your audience in mind) that it is justified. In particular, an obvious statement should be such that a proof of it springs to mind immediately with

[4] According to Steven Pinker, a cognitive scientist, psychologist, linguist, and a popular science author, the biggest reason why many intelligent people write very poorly is "the curse of knowledge". People have a hard time imagining what it is like for someone else to not know something that they know.

no effort. When in doubt about whether a statement qualifies as obvious or not, ask the course staff.

2.6 Should you break up your proof?

When you first learn programming, one thing that will be emphasized over and over again is the value of breaking up your problem into smaller parts and using lots of helper functions. If you write a function that is too long or does more than one task, that is a good indication that you should consider breaking it up into smaller helper functions. Liberal use of helper functions makes your program easier to understand and debug.

In the above sense, mathematical proofs are similar to computer programs. Your proof should be divided up into lemmas and/or claims when appropriate. This will make the proof much easier to understand and it will be easier to spot and fix any potential errors. Even if you are not defining new lemmas/claims, each component of your argument should be separated into different paragraphs, and the high level organization should be easy to identify.

2.7 Does your proof type-check?

In programming, you are used to the idea that every object (piece of data) has a type/class. For instance, many programming languages have an integer type and a string type. The type of the data determines what kinds of operations you can apply on the data. You can, for example, multiply two integers, but you cannot multiply two strings. If you try to do any operation that conflicts with the data type, then you will get a compile-time error, and your code will not run.

Another way to get a compile-time error is by trying to access a variable that has not been defined and initialized. And in statically typed programming languages, you need to explicitly specify the type of a variable when you declare it. This way, the compiler can type-check your program before running it.

Similar to programming, every mathematical object has a type (e.g. it could be an integer, a set, a function, etc.). And as in programming, the type of an object determines what kind of operations you can apply to the object. For example, you can add an element to a set, but you cannot add an element to a function. If you do not respect these constraints, we say that your argument does not type-check. A type-checking error often leads to a nonsensical sentence (even though the author may not realize this).

With these in mind, make sure that your proofs always type-check. You must act as a compiler to find anything that might violate type constraints. In order to make this easier, whenever you learn about a new definition, make a note of the type of the object being defined. This is one of the most important parts of a definition. And when referencing an object in your proof, be aware of its type and make sure that your proof type-checks.

In addition to the above, remember that all the variables in your proofs should be introduced properly. In particular, it should be clear if a variable represents an arbitrary object (e.g. in the context of a "for all" quantifier), a specific object known to exist (e.g. in the context of a "there exists" quantifier), or something else (e.g. a specific value). In all cases, the variable must have a specific type, and it should be clear what the type is.

Here is a suggestion to keep in mind. If in a sentence you are referring to a variable, consider preceding the variable with its type if you feel that this makes things clearer and does not add too much redundancy. For example, if x and y are variables referring to strings, you can consider changing a sentence like "Concatenating x and y..." to "Concatenating string x and string y...".

2.8 Are your references clear?

As you develop an argument in a proof, you will want to refer back to (and use) a previously established statement or an assumption. Implicitly using previous statements or

assumptions can make it hard to follow the logic of the proof and put unnecessary burden on the reader. Therefore, avoid implicit references and make sure it is clear which part(s) of the proof you are using to establish the current step of your proof. Whenever you make a reference, you want it to be very easy for the reader to spot the thing that is being referred to. For example, if there is an important equation that will be referenced later on in the proof, it is a good idea to put that equation on a separate line and label it. This way, you can use the equation's label to refer to it, and the fact that it is on a separate line by itself will make it very easy for the reader to identify it.

In addition to the above, please be careful when you use a word like "it", "this", or "these" in your proof. While you are writing your proof, when you use such a word, you know perfectly well what that word refers. However, ask yourself whether it would also be perfectly clear to the reader. In general, it is a good idea to try to avoid such words as much as possible, even if this means a certain level of repetitiveness is introduced in the used words.

2.9 Where are you using the assumptions?

When you are asked to write a proof, you are usually given certain assumptions that you take as true (which is your starting point), and you have a target statement that you want to derive. As pointed out in the "Are you starting the right way?" section above, it helps you and the reader to explicitly lay out the given assumptions as well as the statement that needs to be derived. Once your proof is complete, ask yourself where in the proof the assumptions are being used. If it turns out that you are not using a given assumption, you should raise your alertness level and check: is the statement even true without that assumption? (There may be some exceptions, but almost always, the answer will be no.) If it is not true, and you do not use the assumption in your proof, then your proof is wrong.

When someone reads your proof to verify it, they will be trying to spot where the assumptions are used in the proof. This is a quick sanity check that the proof is not missing an essential component. In order to make this check easy for the reader (and also for yourself), you should make clear in your write-up where and how the assumptions are being used in your argument. In the rare case that a given assumption is not needed for the proof, mention this explicitly.

2.10 Is the proof idea clear?

As mentioned before, a proof is an argument that convinces the reader that a certain statement is true. After reading the proof, the reader may walk away with a clear and intuitive understanding of why the statement is true. Or, they may not, even if they are perfectly convinced that the statement is indeed true.

We do not want to get into a philosophical discussion about what it means to understand why something is true. But hopefully we can agree that certain proofs have more explanatory content than others.

We strongly encourage you to always write proofs with strong explanatory content! For this reason, if the motivation/intuition behind the proof is not transparent, consider adding a paragraph or two explaining the main ideas that make the proof work. This can be either added within the proof itself, or can be a separate "Proof Idea" section preceding the proof.

Summary.

1. Is the basic structure right?
2. Are you starting the right way?
3. Did you read your proof out loud?

4. Is the purpose of every sentence clear?
5. Are you giving the right level of detail?
6. Should you break up your proof?
7. Does your proof type-check?
8. Are your references clear?
9. Where are you using the assumptions?
10. Is the proof idea clear?

3 An Example

Let's now come back to the example from the beginning of Section 2 and see how it fares. For ease of reference, we reproduce the "proof" here:

Proposed solution.

1. $F_n =$ "$2^n > n$"
2. $F_1 =$ "$2 > 1$" ✓
3. $F_n \implies F_{n+1}$:
4. $2^{n+1} = 2 \cdot 2^n > 2 \cdot n$ (induction) $\geq n + 1$ because $n \geq 1$
5. Therefore proved.

- *Is the basic structure right?* It is quite easy to see that there is a gross violation of this point. The argument does not consist of sentences. In fact, there is not a single fully formed English sentence.
- *Are you starting the right way?* We are missing a restatement of what the given assumptions are and what needs to be derived. (Yes, this is kind of pedantic with the toy example, but still valuable in the context of more complicated examples.) Furthermore, even though the proof is supposed to be a proof by induction, we only learn about this towards the end of the proof.
- *Did you read your proof out loud?* If we attempt to read the argument out loud, we hear an incomprehensible sequence of words.
- *Is the purpose of every sentence clear?* What is the purpose of the first line? Is it a definition? Is it a claim? Is it an assumption? We can also ask these questions for lines 2 to 4.
- *Are you giving the right level of detail?* Implicit in the "proof" is the claim that "$2 \cdot n \geq n + 1$ because $n \geq 1$". This claim is true, and would qualify as an *obviously* true statement. However, there is a step that is being skipped, which if included, would make the claim more immediately obvious. In particular, saying "$2 \cdot n = n + n \geq n + 1$, where the inequality follows because $n \geq 1$." is preferable. Again, in the context of this toy example, our point may come across as too pedantic, but it is good to err on the side of caution and spell things out as much as possible. In another context/proof, a step that you skip might cause the reader a headache.
- *Should you break up your proof?* This proof does not require any "helper functions". It is short and simple enough.

- *Does your proof type-check?* The variable n is not introduced properly. From the problem statement, we can infer that n's type is a natural number. However, the variable should still be declared properly. For instance, F_n is supposed to be the statement "$2^n > n$", but here n is undefined/unquantified.
- *Are your references clear?* There seems to be a reference to an induction hypothesis, but it is poorly presented since there is no indication that the proof is by induction until there is an attempt to make a reference to the induction hypothesis.
- *Where are you using the assumptions?* The part "because $n \geq 1$" is a reference to an assumption that is given by the problem statement, so we expect the author to point this out explicitly. Once again, this point may seem pedantic, but in longer and more complicated proofs, these things really do make a difference.
- *Is the proof idea clear?* As the given problem and its proof are quite simple, a 'proof idea' section is not necessary.

Here is an example of how the proof can be written using good style.

Good solution. We will prove that for all integers $n \geq 1$, $2^n > n$. The proof is by induction on n.

Let's start with the base case which corresponds to $n = 1$. In this case, the inequality $2^n > n$ translates to $2^1 > 1$, which is indeed true.

To carry out the induction step, we want to argue that for all $n \geq 1$, $2^n > n$ implies $2^{n+1} > n + 1$. We do so now. For an arbitrary $n \geq 1$, assume $2^n > n$. Multiplying both sides of the inequality by 2, we get $2^{n+1} > 2n$. Note that since we are assuming $n \geq 1$, we have $2n = n + n \geq n + 1$. Therefore, we can conclude that $2^{n+1} > n + 1$, as desired.