CS103 Handout 11
Fall 2020 October 2, 2020

Guide to Proofs on Discrete Structures

In Problem Set One, you got practice with the art of proofwriting in general (as applied to numbers, sets, puzzles, etc.) Problem Set Two introduced first-order logic and gave you some practice writing more intricate proofs than before. Now that you've hit Problem Set Three, you'll be combining these ideas together. Specifically, you'll be writing a lot of proofs, as before, but those proofs will often be about terms and definitions that are specified rigorously in the language of first-order logic.

The good news is that all the general proofwriting rules and principles you've been honing over the first two problem sets still apply. The new skill you'll need to sharpen for this problem set is determining how to start with a statement like "prove that *R* is an equivalence relation" and to figure out what exactly it is that you're supposed to be doing. This takes a little practice to get used to, but once you've gotten a handle on things we think you'll find that it's not nearly as tricky as it might seem.

This handout illustrates how to set up proofs of a number of different types of results. While you're welcome to just steal these proof setups for your own use, we recommend that you focus more on the process by which these templates were developed and the context for determining how to proceed. As you'll see when we get to Problem Set Four and beyond, you'll often be given new first-order definitions and asked to prove things about them.

This handout is divided into four parts.

- ***The Big Tables***, which explains what you can assume and what you can prove about different first-order logic constructs;
- ***Proof Templates***, which use The Big Tables to show how to structure proofs of definitions specified in first-order logic;
- ***Defining Things***, which explains how to define mathematical objects of different types; and
- ***Writing Longer Proofs***, which explains how to write proofs that feel just a little bit longer than the ones we've done so far.

We recommend that during your first read-through here you proceed in order, then flip through the sections in isolation when you need them as a reference.

The Big Tables

As you've seen in the past week, we're starting to use first-order logic as a tool for writing formal definitions, and the structures of the proofs we'll be writing now depend on how those formal definitions are structured.

Below is a table of all the first-order logic structures we've seen so far, along with what you need to do in order to prove that they're true. Initially, this table might seem a bit daunting, but trust us – you'll get the hang of it!

Statement Form	***Proof Approach***
$\forall x.\ P$	***Direct proof:*** Pick an arbitrary x, then prove P is true for that choice of x. ***By contradiction:*** Suppose for the sake of contradiction that there is some x where P is false. Then derive a contradiction.
$\exists x.\ P$	***Direct proof:*** Do some exploring and find a choice of x where P is true. Then, write a proof explaining why P is true in that case. ***By contradiction:*** Suppose for the sake of contradiction that P is always false and derive a contradiction.
$\neg P$	***Direct proof:*** Simplify your formula by pushing the negation deeper, then apply the appropriate rule. ***By contradiction:*** Suppose for the sake of contradiction that P is true, then derive a contradiction.
$P \land Q$	***Direct proof:*** Prove each of P and Q independently. ***By contradiction:*** Assume $\neg P \lor \neg Q$. Then, try to derive a contradiction.
$P \lor Q$	***Direct proof:*** Prove that $\neg P \to Q$, or prove that $\neg Q \to P$. *(Great question to ponder: why does this work?)* ***By contradiction:*** Assume $\neg P \land \neg Q$. Then, try to derive a contradiction.
$P \to Q$	***Direct proof:*** Assume P is true, then prove Q. ***By contradiction:*** Assume P is true and Q is false, then derive a contradiction. ***By contrapositive:*** Assume $\neg Q$, then prove $\neg P$.
$P \leftrightarrow Q$	Prove both $P \to Q$ and $Q \to P$.

Most of the proofs that you'll write will involve assuming that something is true, then showing what happens as a consequence of those assumptions. The good news is that, like the previous table, there's a nice set of rules you can use to reason about first-order formulas.

Statement Form	***What It Means to Assume This***
$\forall x.\ P$	Should you find any x, you can assume that P is true for that choice of x. ***Important Note:*** One of the most common mistakes that we see around this point in CS103 is not properly distinguishing between what it means to ***assume*** the statement $\forall x.\ P$ and to ***prove*** the statement $\forall x.\ P$. If you are *proving* that $\forall x.\ P$ is true, you pick an arbitrary x and then try to prove that P is true for that choice of x. In other words, you start referring to x as though it's some concrete thing, then see what happens when you do. On the other hand, if you *assume* that $\forall x.\ P$ is true, ***you do not make an arbitrary choice of x and prove something about it***. Instead, when you make the assumption that $\forall x.\ P$ is true, ***just sit tight and wait***. As soon as you have some specific x where it would be helpful to know that P was true, you can pull out $\forall x.\ P$ and conclude that, indeed, P is true. But simply assuming $\forall x.\ P$ won't give you that x to reason about. Think of a formula of the form $\forall x.\ P$ as being like a hammer. It says "if you find a nail , you can use this tool to hit the nail." If I handed you a hammer and said "please hang this picture frame up on the wall," you wouldn't start swinging the hammer around hoping that you'd hit a nail somewhere. That would be recklessly dangerous. Instead, you'd say "okay, let me see if I can find a nail." Then you'd line the nail up where you want it to go, and only then would you swing the hammer at it.
$\exists x.\ P$	There is some actual object x out there that has property P. When ***proving*** an existential statement, you need to do a bunch of work to hunt around and figure out which x will work. But if you're ***assuming*** an existential statement is true, you can immediately say something like "we know there is an x where P is true" and then start using that variable x to refer to that object.
$\neg P$	Negate P as much as possible, then use this table to see what you're now assuming.
$P \land Q$	Assume both that P is true and that Q is true.
$P \lor Q$	Consider two separate cases, one where P is true and one where Q is true.
$P \to Q$	Should you find yourself in a situation where P is true, you can assume that Q is also true. ***Be careful!*** Like the warning in the section on the universal quantifier, if you are assuming that an implication is true, you should not immediately say something like "assume P is true." Rather, you should sit back and wait for an opportunity to present itself where P is true.
$P \leftrightarrow Q$	Assume both $P \to Q$ and $Q \to P$, using the rules above. Or, alternatively, go by cases, one in which you assume both P and Q and one in which you assume both $\neg P$ and $\neg Q$.

Proof Template: Equivalence Relations

Equivalence relations are one of the more common classes of binary relations, and there's a good chance that going forward, you're going to find equivalence relations "in the wild."

Let's imagine that you have a binary relation R over a set A and you want to prove that R is an equivalence relation. How exactly should you go about doing this? As is almost always the case when writing formal proofs, let's return back to the definition to see what we find. Formally speaking, an equivalence relation is a binary relation that is reflexive, symmetric, and transitive. That means that if we want to prove that R is an equivalence relation, we'd likely write something like this:

Proof: We will prove that R is reflexive, symmetric, and transitive.

First, we'll prove that R is reflexive. […]

Next, we'll prove that R is symmetric. […]

Finally, we'll prove that R is transitive. […] ■

Each of those subsequent steps – proving reflexivity, symmetry, and transitivity – is essentially a mini-proof in of itself. In lecture, when we worked with the example of the relation ~ over the set ℤ, we split this apart into three separate lemmas. You can do that if you'd like, though it's not strictly necessary, and for some shorter proofs bundling everything together into one larger proof with clearly-demarcated sections is perfectly acceptable.

The above proof template shows how you'd proceed if you found yourself getting to a point in a proof where you needed to show that a particular binary relation is an equivalence relation. There are several different ways that this could happen. First, you might be given a specific relation and then be asked to prove something about it. For example, you might come across a problem like this one:

Let R be a binary relation over $\mathbb{Z}$ where xRy holds if $x^2 = y^2$. Prove that R is an equivalence relation.

In this case, you've been handed a concrete relation R, and the task is to show that it's an equivalence relation. A proof of this result might look like this:

Proof: We will prove that R is reflexive, symmetric, and transitive.

First, we'll prove that R is reflexive. […]

Next, we'll prove that R is symmetric. […]

Finally, we'll prove that R is transitive. […] ■

Something to note about the above proof: notice that we didn't repeat the definition of the relation R at the top of the proof before then proceeding to show that R is reflexive, symmetric, and transitive. This follows a general convention: if you're proving something about an object that's explicitly defined somewhere, you should not repeat the definition of that object in the proof. Remember – don't *restate* definitions; *use them* instead (see the Discrete Structures Proofwriting Checklist for more information).

As a great exercise, go and fill in the blank spots in the above proof!

On the other hand, you might be asked to show that an arbitrary relation R satisfying some criteria is an equivalence relation. For example, consider this problem, which we solved in lecture:

Prove that if R is reflexive and cyclic, then R is an equivalence relation.

In this case, we should structure our proof like this:

Proof: Consider an arbitrary binary relation R over a set A that is reflexive and cyclic. We will prove that R is an equivalence relation. To do so, we will show that R is reflexive, symmetric, and transitive.

First, we'll prove that R is reflexive. [. . .]

Next, we'll prove that R is symmetric. [. . .]

Finally, we'll prove that R is transitive. [. . .] ■

Notice that in this case, we had to explicitly introduce what the relation R is, since we're trying to prove a universally-quantified statement ("if R is reflexive and cyclic, then …") and there isn't a concrete choice of R lying around for us to use.

Proof Template: Transitivity

Suppose you have a binary relation R over a set A. To prove that R is transitive, you need to show that

$$\forall x \in A.\ \forall y \in A.\ \forall z \in A.\ (xRy \land yRz \to xRz).$$

Remember our first guiding principle: if you want to prove that a statement is true and that statement is specified in first-order logic, look at the structure of that statement to see how to structure the proof. So let's look at the above formula. From the structure of this formula, we can see that

- we ***pick*** arbitrary elements x, y, and z from A (since these variables are universally-quantified);
- we ***assume*** that xRy and yRz are true (since they're the antecedent of an implication); and then
- we ***prove*** that xRz is true (since it's the consequent of an implication).

This means that a proof of transitivity will likely start off like this:

Consider some arbitrary $x, y, z \in A$ where xRy and yRz. We need to prove that xRz. […]

At this point, the structure of the proof will depend on how R is defined. If you have a concrete definition of R lying around, you would likely go about expanding out that definition. For example, consider the relation S defined over $\mathbb{Z}$ as follows:

$$xSy \quad \text{if} \quad \exists k \in \mathbb{N}.\ (x + k = y)$$

Suppose we wanted to prove that S is transitive. To do so, we might write something like this:

Proof: Consider some arbitrary integers x, y, z where xSy and ySz. We need to prove that xSz. Since we know that xSy and ySz, we know that there exist natural numbers m and n where $x + m = y$ and $y + n = z$. […]

Notice that we did ***not*** repeat the definition of the relation S in this proof. The convention is that if you have some object that's explicitly defined outside of a proof (as S is defined here), then you don't need to – and in fact, shouldn't – redefine it inside the proof. Also, although the relation S is defined in first-order logic, note that there is no first-order logic anywhere in this proof, not even in the part where xSy and ySz are expanded into their longer equivalents.

Alternatively, it might be the case that you don't actually have a concrete definition of R lying around and only know that R happens to satisfy some other properties. For example, let's return to the definition of a cyclic relation that we introduced in lecture. As a reminder, a relation R is called ***cyclic*** if the following first-order formula is true:

$$\forall x \in A.\ \forall y \in A.\ \forall z \in A.\ (xRy \land yRz \to zRx).$$

So let's imagine that we know that R is cyclic (and possibly satisfies some other properties) and we want to show that R is transitive. In that case, rather than expanding the definition of R (since we don't have one), we might go about applying the existing rules we have. That might look like this, for example:

Consider some arbitrary $x, y, z \in A$ where xRy and yRz. We need to prove that xRz. Since R is cyclic, from xRy and yRz we learn that zRx. [...]

Notice how we invoked the cyclic property: we didn't write out the general definition of a cyclic relation and then argue that it applies here and instead just said how we could use the statements we began with, plus the cyclic property, to learn something new.

Proof Template: Reflexivity

Let's suppose you have a binary relation R over a set A. If you want to prove that R is reflexive, you need to prove that the following statement is true:

$$\forall x \in A.\ xRx.$$

Following the general rule of "match the proof to the first-order definition," this means that in our proof,

- we ***pick*** an arbitrary element x from A (since this variable is universally-quantified), then
- we ***prove*** that xRx is true (since it's the statement that's being quantified).

A subtle but important point here: notice that in our proof of reflexivity, we don't actually make any assumptions about x other than the fact that it's an element of A. This contrasts with transitivity, where we would make some extra assumptions about the variables we'd chosen because the statement we were interested in proving was an implication. *(Make sure you see why this is!)*

As a result, a proof of reflexivity will often start off with this:

Pick any $x \in A$. We need to prove that xRx. [...]

As before, the next steps in this proof are going to depend entirely about what we know about the relation R. If we have a concrete definition of R in hand (say, if the proof is taking an existing relation that we've defined and then showing that it's an equivalence relation), then we'd need to look at the definition of R to see what to do next. Let's go back to our example of the relation S defined over the set $\mathbb{Z}$ that we defined earlier:

$$xSy \quad \text{if} \quad \exists k \in \mathbb{N}.\ (x + k = y)$$

If we wanted to prove that S is reflexive, we might do something like this:

Consider some integer x. We need to prove that xSx, meaning that we need to show that there's a natural number k such that $x + k = x$. [...]

Take a minute to see why we've structured the proof this way.

Alternatively, it might be the case that you don't actually have a concrete definition of R lying around and only know that R happens to satisfy some other properties. In that case, you'd have to use what you knew about the relation R in order to establish that xRx has to hold.

Proof Template: Asymmetry

Suppose you have a binary relation R over a set A and you want to prove that R is asymmetric. This means you need to prove that

$$\forall x \in A.\ \forall y \in A.\ (xRy \to y \not R x).$$

As usual, we're going to prove this one by looking at the first-order logic statement above to determine what we need to show. The pattern will look like this:

- we ***pick*** an arbitrary elements x and y from A (since these variables are universally-quantified);
- we ***assume*** that xRy holds (since it's the antecedent of an implication), then
- we ***prove*** that yRx is false (since it's the consequent of our implication).

In many ways, this proof structure is similar to that for transitivity – we've picked some arbitrarily-chosen values, made some assumptions about how they relate in some way, and need to show that they relate in some other way as well (here, that $y \not R x$ holds, or, equivalently, that yRx isn't true.) That would give us a proof template like this one:

Let x and y be arbitrary elements of A where xRy holds. We need to prove that yRx does not hold. [...]

This, of course, isn't the only way that you could prove that R is asymmetric. Another option would be to write this as proof by contradiction. If you wanted to do that, you'd *assume* the negation of the above first-order logic statement, which looks like this:

$$\exists x \in A.\ \exists y \in A.\ (xRy \land yRx).$$

If we're ***assuming*** that this statement is true, it means that we're assuming

- there are elements x and $y \in A$ (since we're assuming an existentially-quantified statement) where
- both xRy and yRx are true (since we're assuming a conjunction (and) of two statements).

That would mean that our proof would start off like this:

Assume for the sake of contradiction that R is not asymmetric. This means that there must be some elements $x, y \in A$ where both xRy and yRx are true. [. . .]

From here, we'd be off to the races trying to reach a contradiction.

Proof Template: Not Symmetric

In the previous examples we've seen, we've tried to prove that a relation *does* have some particular property. What happens if you want to show that a relation *does not* have some particular property?

For example, let's suppose R is a binary relation over A and we want to prove that R is not symmetric. For starters, remember that "not symmetric" and "asymmetric" do *not* mean the same thing! Proving that a relation is not symmetric is quite different from proving that it's asymmetric. To see why this is, remember that if we want to prove that a relation is not symmetric, we'd need to prove that the following first-order logic formula is not true:

$$\forall x \in A.\ \forall y \in A.\ (xRy \to yRx).$$

Equivalently, we'd want to prove that the negation of the above formula is true. The negation of that formula is

$$\exists x \in A.\ \exists y \in A.\ (xRy \land y\not R x).$$

If we want to prove that this formula is true, we would show that

- there are choices of x and y (since we're proving an existentially-quantified formula) where
- xRy is true and yRx is false (since we're proving the conjunction (and) of these statements).

There are many ways you could write up a proof of this form. For example, you could write something like this:

Let $x = [\ldots]$ and $y = [\ldots]$. Then xRy is true because $[\ldots]$ and yRx is false because $[\ldots]$.

This one is pretty simple and to the point. We need to show that a certain x and y exist satisfying some criteria, so we just say what choices we're going to make and justify why those choices are good ones. We could also be even more direct than this and omit the names x and y. For example:

Notice that $[\ldots]R[\ldots]$ is true because $[\ldots]$, but that $[\ldots]\not R[\ldots]$ because $[\ldots]$.

The hard part of writing a proof like this is probably going to be coming up with what choices of objects you're going to pick in the first place rather than writing down your reasoning.

Proof Template Exercises: Other Types of Proofs

In the preceding sections, we went over some examples of how you'd structure proofs of different properties of binary relations. However, we didn't cover all the different properties you might want to consider. That's intentional – ultimately, it's more important that you understand how to look at a first-order statement and determine what it is that you need to prove than it is to approach these problems as an exercise in pattern-matching.

Here are a few exercises that you may want to work through as a way of seeing whether you've internalized the concepts from this handout.

- What would a good proof template be for proving symmetry? How about reflexivity?
- Suppose $f : A \to B$ is a function. Give ***two*** proof templates for showing that f is injective.
- Suppose $f : A \to B$ is a function. Give a proof template for showing that f is not surjective.
- A binary relation R over a set A is called ***functional*** if the following is true:

  $$\forall x \in A.\ \forall y \in A.\ \forall z \in A.\ (xRy \land xRz \to y = z).$$

  Write a proof template for showing that a relation R is functional.
- A function $f : A \to A$ is called an ***involution*** if the following is true:

  $$\forall a \in A.\ f(f(a)) = a.$$

  Write a proof template for showing that a function f is an involution.

Defining Things: Binary Relations

It's common in mathematics to need to define a specific function, or a specific binary relation, etc. and then prove it has some specific properties. For example, if you want to prove that it's possible for a function to have a certain behavior, it's common to say "well, here's a specific function I found that happens to have that property." Or you may want to show that a particular equivalence relation has something true about its equivalence classes, in which case having a concrete equivalence relation in front of you to reason about can be helpful.

So let's say you need to define a binary relation. How can you do this?

One option is to define the binary relation *symbolically*, using mathematical notation. If you want to do that, make sure that you tell the reader

- what the name of the binary relation is (or what symbol you'll be using for it),
- what set the binary relation is a relation over, and
- what rule indicates when the binary relation does or does not hold.

For example, I'm personally a fan of this template:

> Consider the binary relation R over the set A defined as follows:
>
> $$xRy \quad \text{if} \quad \underline{\qquad\qquad}.$$

Of course, the names R, A, x, and y are all placeholders, and you don't need to use literally the same wording I have here. Here's an example of what this might look like:

> Let ~ be the binary relation over $\mathbb{N}$ defined as follows:
>
> $$m \sim n \quad \text{if} \quad m^2 + 1 \text{ is a multiple of } n.$$

A note that we wanted to call attention to here: notice that we're using the word "if" here to link the statements "$m \sim n$" and "$m^2 + 1$ is a multiple of n." This might initially seem confusing – wouldn't that be setting up an implication, where we only can say if $m^2 + 1$ is a multiple of n, then $m \sim n$, but there might be other cases where $m \sim n$ holds that aren't covered here?

Fortunately, that's not the case. The English language is confusing in the sense that the word "if" can sometimes be used for implications and sometimes used for definitions. When introducing something new – as we're doing here with the ~ relation – the word "if" means "is defined as" and is not an implication. We'll do our best to point this out to you whenever we use the word "if" just to remove the ambiguity, but eventually it's a good idea to get used to this notation.

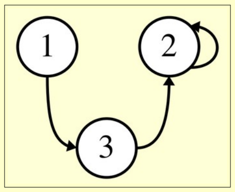
The above template isn't the only way to define a binary relation, and you can be a bit more conversational about this if you'd like. Here's another option:

> Let's have Q be a binary relation over $\mathbb{R}$ where aQb means that $a+b$ is positive.

We still have all the key parts here, and that's all we need.

Another way to define a binary relation would be to draw a picture, like the ones we did in lecture. That's perfectly fine and completely legitimate. The way you might write this in a proof would look something like this:

Consider the binary relation *R* over the set $\{1, 2, 3\}$ as defined by this picture:



[…]

This is a perfectly acceptable way to define a binary relation. We see what the name of the relation is, the underlying set, and the rule that defines it. And for this particular binary relation, which doesn't seem to have any discernable pattern, this might be the easiest way to define the relation!

Regardless of how you choose to specify your binary relations, be careful about making too many claims about them. For example, don't say something like "consider the equivalence relation given below" and then leave it to the reader to confirm that it's reflexive, symmetric, and transitive.

Defining Things: Functions

As with binary relations, there are several ways to define functions. In all cases, you should be sure to do the following:

1. Give the domain and codomain.
2. Give a rule explaining how to evaluate the function.
3. Confirm that it obeys the domain/codomain rules.

Here's an example of what this might look like. Suppose we want to define a function that takes in a natural number, outputs a natural number, and works by adding one to its argument. Here's one way to do just that:

Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined as $f(n) = n + 1$. To confirm this is indeed a function, pick some $n \in \mathbb{N}$. Then $f(n) = n + 1$ is a natural number, as needed.

Notice how we have all the necessary pieces here. We give the domain and codomain using the normal syntax ($f : \mathbb{N} \to \mathbb{N}$), we give a rule for the function ($f(n) = n + 1$), and we check that each natural number (element of the domain) maps to a natural number (element of the codomain).

Sometimes you can define functions with simple rules, where the function is some mathematical expression. Sometimes, you'll need to define functions with more complex rules, where how the function behaves depends on some property of the input. For that, you can use a piecewise function. Here's an example of what that might look like:

Consider the function $f : \mathbb{Z} \to \mathbb{Z}$ defined as follows:

$$f(x) = \begin{cases} x^2 & \text{if } x < 0 \\ x^3 & \text{if } x \geq 0 \end{cases}$$

To quickly confirm that f is a valid function, pick any $x \in \mathbb{Z}$. Then if $x < 0$, we see that $f(x) = x^2$, and x^2 is an integer. Otherwise, we have $x \geq 0$ and $f(x) = x^3$ is an integer.

If you do define a piecewise function, much like when you're writing a proof by cases, the written proof doesn't require you to exhaustively confirm that all cases are covered, but you absolutely should do this on your own to ensure you didn't miss anything!

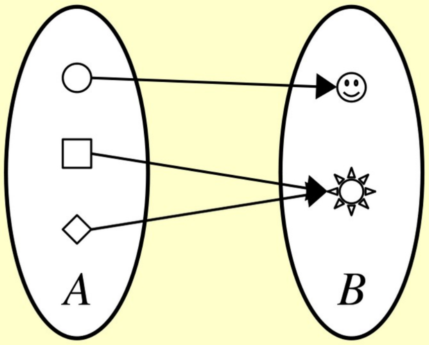
Another way to define a function is to take some existing functions and combine them together. In that case, you don't need to give an explicit rule. For example, suppose that you already have two functions $f : A \to B$ and $g : B \to C$ lying around. Then you could do something like this:

Let $h : A \to C$ be defined as $h = g \circ f$.

At this point, no further elaboration is necessary. The composition operator $\circ$ is well-known and indeed produces a legal function given two input functions whose domain and codomain agree with one another, so there's no need to explain why h obeys the domain/codomain rule. (Stated differently – it's not that there was no need to check that h obeyed these rules, but rather that we already did that outside the proof when defining composition and no further elaboration is needed.)

The last way you can define a function is by drawing pictures showing the domain and codomain and how the function is evaluated. This will only work on functions whose domain and codomain are finite – you can't draw infinitely many items in a picture – and is most common when you're trying to show that it's possible for some function to have some particular property. As with binary relations, you can do this via something as simple as this:

Consider the function $f : A \to B$ defined below:



Here, the reader can see what the sets A and B are and what the rule is for evaluating the function f. This is another exception to the normal rule that you have to validate the domain/codomain rule, since in the case of a function the expectation is “hey reader, I assume you’ve checked there’s exactly one arrow coming out of everything in A, and I’m not going to explain this to you.”