Notes on

Proofs

by Charles Lin

1 Why Study Predicate Logic?

At this point, you should have done plenty of proofs in propositional and predicate calculus. You may find it weird to do such proofs. After all, they only prove such statements as $\forall x\ [\, P(x)\,]$ true. What does such a statement mean anyway?

For some of you, the proofs weren't so bad. Even if what you were proving had little meaning, doesn't integrating and differentiating have much meaning either? Sure, they tell you that you are computing the area under a curve, or computing a slope, but what does that really mean anyway? Until you take courses in engineering or physics, where those functions have some meaning, it may be hard to grasp why you take calculus.

Neverthless, once you see the patterns, you can think of solving calculus problems like solving any other math problems—as a kind of intellectual puzzle. And why solve such puzzles? Because in a strange way, it is teaching you to think,

However, there is a reason to study predicate logic, and that's to understand the structure of proofs. As you already know, a proof is a sequence of steps starting from premises (your assumptions) and working your way to a conclusion, using proof rules and logical equivalences.

Proof rules and logical equivalences are said to be "sound" rules. That means, using proof rules and logical equivalences, you can only prove statements which are logical consequences of your assumptions. As long as the assumptions aren't contradictory, you won't prove a contradictory statement. (Even proof by contradiction, doesn't really prove a contradiction. Instead, it allows you to assume a fact, within a subproof, which causes a contradiction to occur).

Logic is such a big topic that a few weeks worth of study doesn't even begin to touch on all the important issues. It's common for math departments (even computer science departments!) to have a semester long course in logic. Neverthless, you have enough background in proving logical statements to understand the basic structure of a proof. That's the goal.

To understand where this is leading, I want to summarize part of the plot of a semi-forgettable film called *The Karate Kid*, which tells the story Daniel, who has moved out to California with his single mother, and is having some troubles adapting to his new surroundings, especially due to the presence of bullies who, surprise, are karate students. After getting in a fight and being rescued by the sage Mr. Miyagi (veteran actor, Noriyuki "Pat" Morita), he requests that Mr. Miyagi teaches him karate.

Mr. Miyagi's training seems to consist of making Daniel do all sorts of chores, including painting the fences, learning to wax, and so forth. Such tasks have Daniel doing chores all

day long, to his increasing frustration that he isn't really learning any karate, and leads to a key scene where he gets angry at Mr. Miyagi, feeling he's being treated as some sort of free labor. Of course, Mr. Miyagi demonstrates that all the movements he's been using to paint and wax have all been really karate moves in disguise, and so he has, in a way way, been learning karate.

Similarly, your study of logic has been to prove predicates like $P(x)$ and $Q(x, y)$ and all sorts of combination. These predicates were basically meaningless. But that was part of the point! Suppose someone told you p is true and $p \to q$ is true. Can't you conclude that q is necessarily true by *modus ponens* regardless of what p and q *really* stand for?

The study of logic is really the study of logical *form*. If you see something written as p and $p \to q$, then concluding q is based on the *form*, rather than what p and q stand for. So, if someone says "If ducks quack, then the stock market will go up" and then says "Ducks quack", then you can conclude "the stock market will go up". At least, you can conclude this *assuming* the two statements said earlier were true. You might argue that the statements aren't true. (Is "If ducks quack, then the stock market will go up" even true?) However, logic is more concenrned with proving true statements given that other statements (premises) are true, not with determining whether the premises are true.

If you studied "real" math proofs right away, without knowing the structure of what a proof looks like, you might be confused as to what you can and can't do in a proof. In some ways, proving statements in propositional/predicate logic (the kind you've been doing so far) is like studying scales in music. A pianist, for example, might play scales over and over to get practice on scales, even though people don't go to see pianists play scales. When a pianist then plays a piece, they can see the structure of the piece based on the scales they've played. Now that you've had enough practice playing "scales", you're ready to do "real" math proofs.

1.1 The Important Proof Rules

As you've been learning to prove statements in predicate logic, there are a few proof rules and proof strategies that are important to remember when it comes to doing real math proofs. Here are a list of them.

Existential instantiation When you have a statement such as: $\exists x\, [\, P(x) \to Q(x)\,]$, you are allowed to pick an unused constant, such as b, and *instantiate* the statement. For example, you can instantiate it to: $P(b) \to Q(b)$.

What does existential instantiation mean? Suppose someone says "There's someone in this class who score the highest on the exam". Existential instantiation says "let's give that person a 'name', such as Y". Now, Y is only a hypothetical name, but we can then say, for example, that Y received an 'A' on the exam, and that Y has a good chance of passing the course, and that Y seems to have a very good understanding of the material, and perhaps Y should be considered for an honors certificate, even as we don't really know who Y really is. That's the basic idea of existential instantiation. Give some (new and unique) name to the variable.

Existential addition Existential instantiation is used when you have either been given an existential statement (such as $\exists x\ [\, P(x) \rightarrow Q(x)\,]$) or you've proved it, and now you want to pick a new constant.

Existential addition is the opposite. You don't have an existential statement, but you try to find a constant that satisfies the statement. For example, if you finally prove $P(b) \rightarrow Q(b)$, you can now claim $\exists x\ [\, P(x) \rightarrow Q(x)\,]$.

Universal addition Universal addition is the only proof rule specific to predicate logic which requires an assumption. The idea of universal addition is to assume the existence of some constant in the domain, then prove something about that constant, then say that there was nothing special about that constant, and therefore what holds true for that constant holds true for every element in the domain. This is also called a *generic particular* argument, saying that if it's true for some generic element in the set, it's true for all elements.

Generic particular arguments are used all the time in infomercial ads for, say, weight loss. You see a bunch of people who claim they lost many pounds based on some diet plan or exercise regime. They are like "generic particulars". They are implying "if some average person like me can do it, then so can you".

$\rightarrow$ **Addition** When trying to prove something of the form $\alpha \rightarrow \beta$, you assume α, then prove β. This is a basic strategy used in proving conditionals. (You could also do a proof by contradiciton by assuming α and proving $\alpha \wedge \sim \beta$, but it turns out in math proofs that a straight-forward strategy of "assuming α and proving β" usually works).

$\leftrightarrow$ **Addition** When trying to prove something of the form $\alpha \leftrightarrow \beta$, you actually do two subproofs. The first proof is to prove $\alpha \rightarrow \beta$ by assuming α and proving β. The second proof is to prove $\beta \rightarrow \alpha$ by assuming β and proving α.

Proof By Contradiction This has to be the most widely used technique of proving anything that isn't a conditional statement. Without proof by contradiction, you'd be stuck with many proofs. The idea is the same as it was in predicate logic or propositional logic.

You are attempting to prove a statement α. However, you assume the fact $\sim \alpha$. If α is indeed true, then the assumption $\sim \alpha$ will eventually contradict α. Usually, if α is true, then assuming $\sim \alpha$ will lead to other contradictions first. And, because adding $\sim \alpha$ leads to contradictions, then it must be the case that α was true.

Sometimes people use proof by contradiction in real life. For example, someone might say "how come you went to the bar last night", to which the reply might be "I wasn't at the bar last night. If I were at the bar, then how do you explain that I was at so-and-so's place. I can't be at their place and at the bar, right? So, I wasn't at the bar last night". This is proof by contradiction.

You will see many of these techniques show up when we do math proofs.

1.2 Why Not Use Predicate Logic?

As we do "math proofs", you will discover that the proofs are usually written in a stylized English (more of a "math" English), rather than defining predicates. You won't see us using the 'T' method (or the 'E' method), nor justifying each and every step, in a very rigorous manner.

Why not? Here's an analogy. Suppose you wanted to describe someone how your program works. Would they rather see every single line of code you've written, or would they rather hear you describe the "pseudocode", i.e., a high level description of what your program does? More than likely, they would rather hear a high-level description rather than be bogged down by way too many details.

Similarly, a predicate logic proof is like reading a program. You have to include every tiny detail, otherwise the proof is wrong or incomplete. A programmer wastes many hours trying to get every last detail to work, even as they have basically the correct idea when they write the code. Mathematicians don't want to expend that much effort to get every single little detail correct, particularly since such details have often been proven correct before. Obviously, you need to be careful when writing proofs, just like a person needs to be careful describing how a program will work once they get down to writing it.

For example, what can you conclude given that $x < y$ and $y < z$. You can conclude that $x < z$. You can't prove this in predicate logic. Predicate logic says nothing about what $<$ (less than) means. (No, this is NOT an example of hypothetical syllogism. In particular, there are no $\rightarrow$ which appears. It is similar to hypothetical syllogism, but it is not hypothetical syllogism).

What you have to do is to *define* a predicate for less than. For example, that might be $L(x, y)$ to stand for x is less than y. Then, you would have to state an "axiom" or some statement that you want to be assumed true. In particular, you want to state that $\forall x, y, z\ [\, L(x, y) \wedge L(y, z) \rightarrow L(x, z)\,]$. In fact, any reasonable mathemetical "fact" would have to encoded in logic. As you might imagine, having to write out all facts of math in logic is rather painful, and the proofs could be potentially very long, and perhaps far more complicated than it has to be.

So, instead of writing proofs in such ways, we merely structure the proofs based roughly on the predicate logic proofs. What do you gain from doing predicate logic proofs? Mainly, you know when you can and can not assume various facts, and when to introduce new constants. These are all based on the proofs rules listed above.

2 A Mini-Essay on Why Proving is Tough

Why are proofs so tough? Why do students continuously struggle with proofs? I believe part of the problem is that we don't have anything like a compiler for writing proofs. A compiler, in its way, is a great teacher. Why? Because it constantly tells you when you are wrong. At least, it tells you when you're wrong syntactically. Then, from then on, you can observe the outputs of your program with test inputs, and see if your program behaves as expected.

Proofs don't work quite that way. There's no test input; there's no compiler; there's nothing to tell you whether the proof looks good or bad. Even though students are presented with "good examples" of proofs in the book or in class, when asked to prove exactly the same proofs as the ones in the book or class, students routinely write their proofs that are missing many steps, which means the proofs are less than accurate, or even potentially wrong.

This leads me to a controverial point. I believe you need to "memorize proofs". I can't quite say how well you need to memorize it. Do you need to memorize it like you memorize the Gettysburg Address? Do you need to know it word for word, and if any one word is wrong, then well, you just don't know the proof? I want to say, "of course not, you don't need to memorize it to that detail".

And yet, if you don't memorize it to that detail, then you're likely to not be anywhere near close. Let me give you an analogy. Suppose I asked you to memorize Gettysburg Address (which, by the way, is one of Abraham Lincoln's most famouse speech, given sometime during the Civil War), but you didn't have to memorize every single word. How far would you get? I think most computer science students might get as far as "Four score and seven years ago", which is about as much as anyone typically has memorized of the speech. However, there are several hundred more words that are missing, and so to say the first six words accurately represent the entire speech would be a huge mistake.

I believe the typical computer science major would do the equivalent of memorizing six words when "memorizing a proof" that they didn't have to memorize.

Let me tell you why memorizing is considered controversial. Somehow memorization is equated with lack of understanding. I completely disagree with this. I believe memorization is on the way to understanding. If you don't memorize a few things, how can you expect to understand? It would be incorrect of me to say "I understand the $\sqrt{2}$ proof, but I didn't memorize it". Of course, I memorized it! But I also understand it well enough to do proofs of $\sqrt{5}$ proof as well, and can show why $\sqrt{4}$ proof would work incorrectly if I used the same same technique.

I'll tell you why it's less controversial than you might imagine. Pretend your sitting in class. You hear me say "I strongly recommend you to memorize the proofs". Would you then tell yourself "Hmm, he's right, I better memorize that proof, because it will be important for me to do so"? Why do I believe you would do nothing of the sort?

I'll tell you the answer. Because students are used to only doing a handful of things when it comes to class. If it becomes too unusual based on their experience, they won't do it. For example, students are used to the idea of doing homework, typing up papers and essays, and after some time, learning how to write programs. Suppose I ask you, in class, please go to a nearby museum and take a picture of a sculpture or painting, and bring it to class next lecture. How many students do you think would do this? I would say "about 0". Why?

Because taking a picture seems way outside the bound of what is reasonable for one to ask a student to do (although art students may be asked that). There are many things that make it difficult. Finding a museum, getting a camera, getting the picture developed in time, finding the time to do all of this. It's simply easier to think "they don't seriously want me to do this, so I won't do it".

This is one reason that learning is so difficult for students. Professors are often suggesting many ideas for students, and yet they never take heed of most of the advice ("please start on your homework early", "make sure you do the practice problems", "if you have problems, come into office hours"). All of this is routinely ignored. Sometimes, it's ignored with semi-valid excuses ("I have other things to do, other courses I have to deal with").

However, if you want to master proofs, the advice I give is to try to impersonate proofs done in class and done in the book and constantly ask yourself "What's wrong with the way I wrote my proof? How is it different from the one done in class?" because, unless you have a strong math background with proofs, you are almost certainly doing something wrong. This presupposes that you will actually attempt to do proofs.

Let me repeat something rather obvious to you: *In order to learn how to prove, you must prove*. I suggest doing the proofs in the book and class over and over so that if you were called in front of the class, you could actually do the proofs. In fact, I believe that the best way to learn proofs is to get up to a board and present it to your friends.

If you aren't practicing proofs over and over and over again, then you will never get used to writing proofs. The first place to start is to do exactly the proofs done in class. Are you listening? Are you planning to do those proofs?

3 Definitions!

Suppose you are asked to prove something for a homework problem. In the proof, the word "rational" is used, or perhaps you see $p \mid q$, and realize that this is the "divides" symbol. What should your first thought? It should be the actual definition of the words or symbols used. For example, if you see the word "rational", you should be thinking of the *definition* of rational. If you see the symbol $\mid$ in $p \mid q$, you should be thinking of the definition of divides.

This leads me to my first piece of advice when learning proofs.

> ***Advice**: When learning proofs, memorize the definitions of terms, then apply the definition.*

Do you really, really, really need to memorize definitions? Yes. (OK, you don't, but remember the "Gettysburg Address" story. If you don't memorize, will you even try to get anything close?).

Here are a list of definitions that you should know. Fortunately, there's only seven of them. All of these definitions are written in predicate logic. Of course, you need to be able to translate them to English (and back). Part of writing a definition is knowing how to write it in prediate logic correctly and how to write it in English.

rational

$$\text{rational}(x) \iff \exists a, b \in \mathbb{Z}\,[\, x = \frac{a}{b} \wedge b \neq 0\,]$$

In English, a number x is rational if you can write it as a fraction. Even though this definition doesn't say it, x is not uniquely written. For example, suppose x is 1/4, then it can also be written as 2/8, 3/12, 4/16, and so forth. Despite an infinite number of ways of writing 1/4, they all represent the same rational number. Thus, even though 1/4 and 2/8 are written different (i.e., their written *representation* is different), the actual meaning (i.e., the actual number) it refers to is the same. For example, if you had written a program and wrote the assignment statement $x = 1/4$ and $x = 2/8$, more than likely, the program would store the same number in memory (it's base 2 floating point representation).

prime The predicate, prime(n), is defined on integers. That is, $n \in \mathbb{Z}$. More precisely, it's defined on integers greater than 1.

$$\text{prime}(n) \iff [\, n > 1 \wedge \forall a, b \in \mathbb{Z}^+ \, [\, n = ab \rightarrow a = 1 \wedge b = 1 \,]$$

In English, a number is prime if it is greater than 1, and if that number is a product of a and b where one of the values is 1 (notice that both can't be 1, because that could make $n = 1$, and we have $n > 1$ to make sure that doesn't happen).

Notice that 1 is not considered prime (and neither is 0, nor any of the negative integers). Obviously, numbers that aren't integers can't be prime.

Also notice that a and b are picked from positive integers. (This would be an easy mistake to make when memorizing the definition).

composite Like prime(n), the predicate, composite(n), is defined on integers greater than 1.

$$\text{composite}(n) \iff [\, n > 1 \wedge \exists a, b \in \mathbb{Z}^+ \, [\, n = ab \wedge a \neq 1 \wedge b \neq 1 \,]$$

In English, a number is prime if it is greater than 1, and if that number is a product of a and b where one of the values is 1 (notice that both can't be 1, because that could make $n = 1$, and we have $n > 1$ to make sure that doesn't happen).

Notice that 1 is not considered prime (and neither is 0, nor any of the negative integers). Obviously, numbers that aren't integers can't be prime.

even The predicate, even(n), is defined on integers (positive and negative and zero).

$$\text{even}(n) \iff \exists k \in \mathbb{Z} \, [\, n = 2k \,]$$

We will give an alternate definition later on.

odd The predicate, odd(n), is defined on integers (positive and negative and zero).

$$\text{odd}(n) \iff \exists k \in \mathbb{Z} \, [\, n = 2k + 1 \,]$$

We will give an alternate definition later on.

divides The "divides" predicate has a symbol, namely $|$, which is a two parameter predicate. Thus, you write $p \mid q$ to say "p divides q". p and q are

$$\forall p, q \in \mathbb{Z} \, [p \mid q \iff \exists k \in \mathbb{Z} \, [\, q = pk \,] \,]$$

Notice that I write a $\forall$ for p and q, which did not appear in any of the previous definitions. They should have appeared, but it's convenient not to always write it out. In this case, however, I want to quantify p and q to specifically illustrate which domain p and q are part of (namely, $\mathbb{Z}$).

congruence Congruence is a predicate that has three parameters: a, b and n. It's written as $a \equiv b \pmod{n}$.

$$\forall a, b \in \mathbb{Z} \, \forall n \in \mathbb{Z}^+ \, [\, a \equiv b \pmod{n} \iff \exists k \in \mathbb{Z} \, [\, a = nk + b \,] \,]$$

Again, the $\forall$ that appears is useful for saying which domain a, b and n are in. In particular, a and b are any integers while n is a positive integer. Another way of writing this definition is:

$$\forall a, b \in \mathbb{Z} \, \forall n \in \mathbb{Z}^+ \, [\, a \equiv b \pmod{n} \iff n \mid (a - b) \,]$$

This is an equivalent definition, but it uses divides. How equivalent is it? If you apply the definition of divides, you will get:

$$n \mid (a - b) \iff \exists k \in \mathbb{Z} \, [\, a - b = nk \,] \,]$$

With a little algebra, $a - b = nk$ can be rewritten as: $a = nk + b$, which is the same as the previous definition.

3.1 Alternate definitions

Now that divides and congruence mod n have been defined, then there are alternate definitions. For the most part, these alternate definitions are exactly equivalent to the ones given. Occasionally, however, there will be definitions that are "equivalent", but it would take a lot more math to show that they were equivalent.

even The predicate, even(n), is defined on integers (positive and negative and zero).

This alternate definition uses divides.

$$\text{even}(n) \iff 2 \mid n$$

This alternate definition uses congruence.

$$\text{even}(n) \iff n \equiv 0 \pmod{n}$$

Both definitions are basically the same as the original.

odd The predicate, odd(n), is defined on integers (positive and negative and zero).

This alternate definition uses divides.

$$\text{even}(n) \iff 2 \mid (n+1)$$

This alternate definition uses congruence.

$$\text{even}(n) \iff n \equiv 1 \pmod{n}$$

Again, both definitions are about the same as the original.

divides "divides" can also be written using congruence.

$$\forall p, q \in \mathbb{Z} \left[\, p \mid q \iff q \equiv 0 \pmod{p} \, \right]$$

If you just apply the definition of congruence and the original definition of divides, you will see that the two are equivalent.

3.2 Understanding a Definition

When trying to master the definition, you need to first get familiar with the symbols used and how to use it. In particular, students always get confused with $\mid$ (i.e., divides) and $\equiv$ (i.e., congruence).

For example, when reading $3 \mid 12$, what do you think? The symbol looks weird. How do you even read it aloud?

You read it aloud one of two different ways. Either you say "3 divides 12" (which is more accurate) or you say "12 is divisible by 3". Notice that you write $3 \mid 12$, rather than $12 \mid 3$. This may seem unusual, based on your knowledge of division, but that's the way "divides" works.

What's the difference between $12/3$ and $3 \mid 12$? The first is simply a division, and gives a number as the result. If someone asks you, what is $12/3$, you would answer 4. However, if someone asks $3 \mid 12$, the answer is "true". That's because $3 \mid 12$ means "12 is divisible by 3", which is either true or false. $\mid$ is a predicate, meaning that it is a function that returns either true or false.

It turns out that $-3 \mid 12$ too. That is, 12 is divisible by -3. And $-3 \mid -12$ and $3 \mid -12$ are all true. You might wonder why. This is the next step. As you are learning a new operator, you first learn what it means "intuitively", which means you are able to compute with it. Frequently, students don't practice and understand what the operator means. This is like learning a foreign language but never practicing. You will never get very good at it.

However, the next step is to *know* the definition, which basically means to memorize the definition. The reason? When you start to do proofs, you really want to use the definition as listed above, and not rely on what you sort of think it is, but you're not 100% sure.

Let's consider an example. How would you *really* show that $3 \mid -12$? You apply the definition of divides. Recall the definition:

$$p \mid q \iff \exists k \in \mathbb{Z} \left[q = pk \right]$$

Substitute 3 for p and -12 for q and you get:

$$3 \mid -12 \iff \exists k \in \mathbb{Z} \left[-12 = 3k \right]$$

The goal is to find a value of k such that $-12 = 3k$. Divide both sides by 3 and you get $k = -4$. Since there does exists an integer, k, (namely, when $k = -4$), then $3 \mid -12$. When you're unsure of what to do, you need to go back to the definition and apply it.

Let's see if $12 \mid 3$.

$$12 \mid 3 \iff \exists k \in \mathbb{Z} \left[3 = 12k \right]$$

If you solve for k, you get $\frac{1}{4}$, which is not an integer. Therefore, there does not exist an integer k such that $3 = 12k$, and therefore $12 \not| 3$. This makes sense too. 3 is not divisible by 12 (at least, not as a whole number).

If you're having problems deciding whether $p \mid q$ for some value of p and q, then use the definition and see where it leads you.

Congruence is the other operator that seems to give students a lot of problems. What do you think when you see something like: $a \equiv b \pmod{n}$? Many students, recalling the `mod` operator from C/C++ (which is written as %), read the statement as:

$$a \equiv b \% n$$