Notes on
Predicate Calculus

by Charles Lin

1 Propositional Calculus Enough?

Propositional calculus is good enough to describe certain kinds of reasoning, but it is lacking. For example, consider the following argument.

Every computer science major likes logic.
Tina is a computer science major.
___
∴ Tina likes logic

Do you believe that this is a valid argument? It seems like it should be, but we'll see if this can be proved within propositional calculus. As usual, the first step of proving this argument is to convert the sentences to statement letters. For example, we might use:

p: Every computer science major likes logic.
q: Tina is a computer science major.
r: Tina likes logic.

This would lead to the translation:

p
q
___
$\therefore r$

There's no rule (other than a proof by contradiction, with contradictory premises) that will allow you to conclude r when r doesn't appear at all in the premises.

We might even try to use a more appropriate translation. For example, we could let p stand for "Someone is a computer science major" and s stand for "Someone likes logic", which would allow the first sentence to be translated to $p \to s$. Even so, we could not conclude, r.

Even though we may feel that "Someone is a computer science major" should be related to "Tina is a computer science major", there is nothing in propositional calculus that allows us to relate the two, and so they have to be translated as different statement letters.

2 The Need For Predicates

Predicate calculus allows to add predicates to the language. What's a predicate?

Definition *A **predicate** is a Boolean function, which evaluates to true or false*

With predicates, we can make progress towards a proof of the previous argument.

$C(x)$: x is a computer science major. $L(x)$: x loves logic.

With these predicates, we can begin to say sentences like $C(Tina)$ to mean “Tina is a computer science major” and $L(Tina)$ to mean “Tina loves logic”. However, we’ll need more than just predicates. So, let’s begin by introducing basic set theory, then the PRED language.

3 Basic Set Theory

To understand predicate calculus, we’ll need to make a little detour to basic set theory. We’re just going to cover enough set theory so you can understand predicate calculus. This is not meant to be an in-depth treatment of set theory.

What’s a set?

> **Definition** *A **set** is a collection of objects*

This isn’t a very satisfying definition, but typically this is how a set is defined. In math, at some point, one has to rely on the person reading the definition to understand what it means, and one hopes that a set is a basic enough concept for that.

One way of writing set by using an open brace, then listing the elements, and separating it by commas.

$$\{2, 4, 6, 8, 10\}$$

The elements or members of the set are: 2, 4, 6, 8, 10. You can give the set a name as well.

$$\mathcal{S} = \{2, 4, 6, 8, 10\}$$

In this case, we give the name $\mathcal{S}$ so that it’s not necessary to keep writing $\{2, 4, 6, 8, 10\}$.

3.1 Membership is a Fundamental Property of Sets

Perhaps the most fundamental property of sets is membership. Either an element is in the set, or it is not. We write membership using the symbol, $\in$. Thus, once can say $2 \in \mathcal{S}$, which is read as “2 is a member of the set $\mathcal{S}$”. The symbol $\notin$ is read as “is not a member of”. Thus, $3 \notin \mathcal{S}$ which is read as “3 is not a member of the set, $\mathcal{S}$”. It’s also common to translate $2 \in \mathcal{S}$ as “2 is an *element* of the set $\mathcal{S}$”.

Most of the times, we will talk about sets of numbers, but sets can contain anything, from people’s names, to lists of programs, to student ID numbers. You can even have completely unrelated items in sets, putting numbers, names, programs—whatever you can think of, in a set. Practically speaking, most people use sets of numbers or sets of things that are mathematical in nature.

Since membership is such a key property, we won't care about other issues such as how often an element appears in a set. All that's important is whether the element is in the set or not. Therefore, the following set:

$$\mathcal{S} = \{4, 4, 2, 2, 2\}$$

is exactly the same set as

$$\mathcal{S} = \{2, 4\}$$

The order of the set doesn't matter, and how often an element appears doesn't matter. Two sets are equivalent if both have exactly the same members.

Even though it doesn't matter how often an element appears in a set, we will follow the convention of writing a set so that each element only appears once. Thus, if you are asked to write a set with 2 and 4 as its only elements, you'd write it as $\{2, 4\}$ or $\{4, 2\}$, but not as $\{2, 2, 2, 4\}$. There is something called a *bag* (also known as a *multiset*) where you can not only ask whether an element is in the bag or not, you can also ask how often it appears in the bag. Thus, bags keep track of both membership and frequency. Sets only keep track of membership.

3.2 Alternate Notation: Truth Sets

One way to write out a set is to "enumerate" the entire set, i.e., write out each element of the set. While this is fine for small sets, it becomes very unwieldy for large sets, and impossible for infinite sets. Thus, another way to describe sets is to describe a property that each element of the set has. This is usually written as:

$$\mathcal{S} = \{x \mid x \text{ has property } P\}$$

Thus, we say "Set $\mathcal{S}$ contains all elements x such that x has property P." The vertical bar that appears in the set above is translated as "such that".

Here's an example:

$$\mathcal{S} = \{x \mid x \geq 2 \wedge x \leq 100\}$$

Set $\mathcal{S}$ therefore contain all elements that lie between 2 and 100.

This notation allows us to write infinite sets as well.

$$\mathcal{S} = \{x \mid x \geq 2\}$$

As precise as this way of specifying sets may be, it actually is ambiguous. The problem? We never said what kinds of values x could assume. Is x supposed to be integers, reals, complex numbers? To talk about numbers, we need to make a list of well-known infinite sets.

3.3 Well-known Infinite Sets

Here are a list of well-known infinite sets.

$\mathbb{Z}$ The set of integers (why $\mathbb{Z}$ is picked, I'm not sure—probably integers start with 'Z' in another language). The set of integers is often written as $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

$\mathbb{Z}^+$ The set of positive integers (i.e., integers greater than 0).

$\mathbb{Z}^-$ The set of negative integers (i.e., integers less than 0).

$\mathbb{Q}$ The set of rational numbers (numbers which can be written as p/q where p is any integer and q is a non-zero integer). Notice that all integers are also rational numbers.

$\mathbb{R}$ The set of real numbers. These are the set of rational and irrational numbers. Irrational numbers are numbers which can't be written as fractions. They are basically, not repeating decimal numbers. Perhaps the most famous irrational number is π. The second most famous irrational number is probably $\sqrt{2}$. (The third is probably e which is roughly 2.71). Even though irrational numbers form their own set, there's no commonly used letter used to represent the set of irrational numbers.

$\mathbb{C}$ The set of complex numbers. Complex numbers are written in the form $a + ib$ where a and b are both real numbers and i is defined as $\sqrt{-1}$.

Now that we can use "well-known" sets, we can write the set used earlier more precisely as:

$$\mathcal{S} = \{x \in \mathbb{Z} \mid x \geq 2\}$$

This is the set of all x picked from the set of integers, such that x is bigger than 2. In other words, this is the set of all integers greater than 2.

3.4 Summary

Thus, you can write sets one of two ways:

1. Enumerate each element (i.e., write out all the elements of the set)
2. Use truth sets, and describe the sets using properties which each element of the set has

4 The PRED language

In predicate calculus, as with any logic, it becomes convenient to describe what makes a valid formula. We can think of this as a language, much like a computer language, except that it's usually much easier to understand PRED than a normal language. At least, it's easier to determine whether a formula in PRED is syntactically valid or not.

Predicate calculus has more symbols than propositional calculus, which makes it more confusing. So, by explaining the language, you should be aware of the various parts of the language. We call it a "language" in much the same way that people call, say, Java a language. It's not a spoken, natural language like English or Japanese. Instead, it's a mathematical language, described by precise rules. Fortunately, because you have seen programming languages like C or C++ before, the terminology used to describe predicate calculus shouldn't be so bad.

4.1 Symbols used in PRED

Here are a list of symbols used in predicate calculus.

lowercase letters Lowercase letters will stand for either variables (which usually appear in the later parts of the alphabet, such as w, x, y, z) and constants (which usually appear in the early part of the alphabet such as a, b, c, d). These letters can have subscripts. If I wanted to be completely formal and precise, I'd only permit variables to be the letter x with subscripts, and constants to be the letter c with subscripts. However, that would make various formulas difficult to read. So, I'll be somewhat informal, and let you know when a lowercase letter is a variable or a constant.

uppercase letters Uppercase letters will stand for predicates. Recall that a predicate is a Boolean function. A $n - place$ predicate is a Boolean function with n parameters You can have as many (finite) parameters as you want, and as few as 0 parameters. Once it's been decided how many parameters a predicate has, then it is fixed for the duration of the current problem. Predicates can also have integer subscripts.

Again, it will be convenient to be informal about predicates. For example, we might write predicates using descriptive English words. Thus, even(x) can stand for the predicate, x is an even integer.

calligraphic uppercase letters These will be used to represent sets. For example, $\mathcal{S}$ would be a set.

the "element of" symbol as in the symbol, $\in$.

quantifiers There are two quantifiers: $\forall$ (read as "for all", and written as an upside down 'A') and $\exists$ (read as "there exists" or "there exists at least one" and written as a backwards 'E')

parentheses I will use parentheses (,), as well as brackets [,].

We'll discuss what a quantifier is later on.

4.2 Well-formed formulas

Now that we have our symbols, we can define syntactically valid logic formulas. These will be called *well-formed formulas*. One could write a compiler which would read in a formula and tell you whether it was well-formed or not. Only well-formed formulas (wff's for short) have meaning. Formulas which are not well-formed are supposed to be nonsense, just like a program that doesn't compiled is considered "nonsense". However, we'll often be informal and allow non-wff's, as long as they appear reasonably close to wff's.

1. All n-place predicates with the n parameters are wff's. Each of the n parameters must either be a constant or a variable.
2. If α is a wff, then so is $\sim \alpha$.
3. If α and β are wff's, then so are $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.
4. If α is a wff, x is a variable, and $\mathcal{D}$ is a set, then $\forall x \in \mathcal{D}(\alpha)$ and $\exists x \in \mathcal{D}(\alpha)$ are wff's.

If there's only one domain (and typically, there is), then sometimes, I will use the more informal: $\forall x(\alpha)$ and $\exists x(\alpha)$ and not refer to the domain at all, leaving it implicit.

5 What Are Quantifiers?

Suppose you wanted to translate a sentence like: "Everyone loves logic". Since we're studying predicate logic, we can use a predicate to describe the property of "loving logic". Let's do that.

Let $L(x)$ stand for "x loves logic"

But how can we say "Everyone loves logic"? And what does it really mean to say "Everyone" anyway? How can we say "Someone loves logic"?

Predicate calculus gives us two quantifiers to express the idea of "everyone" and "someone". The first quantifier is $\forall$ which is called the *universal quantifier* and is often read as "for all". It is written as an upside down 'A'. The second quantifier is $\exists$ (written as a backwards 'E'), and is the *existential quantifier* and is often read as "there exists" (but is more accurately read as "there exists at least one").

Thus, to say, everyone loves logic, we want to use the universal quantifier $\forall$. This is the way it is typically written:

$$\forall x \in \mathcal{D} \ \ (L(x))$$

It's usually read out loud as "for all x in a domain D, L of x". x is a variable, and we are letting this variable take all values of the elements in the domain, D. The domain is a set, and will let us define what we mean by "everyone". What it really means is "for every element in the set D". Thus, if set D represents "the set of all people", then $\forall x \in \mathcal{D}\ \ (L(x))$ can be translated into English was "Everyone loves logic".

Admittedly, "the set of all people" can be ambiguous too, since we can't claim for certainty, what that really means. Fortunately, we will eventually use predicate calculus to prove mathematical theorems, and the domains will become more precise.

The other quantifier which we can use is $\exists$, which will allow us to translate "There exists (at least one) person who loves logic".

$$\exists x \in \mathcal{D}\ \ (L(x))$$

Notice that I used the phrase "at least one", which leaves open the possibility that there could be more than one (and possibly even everyone).

How do we say "Every computer science major loves logic"? Would it be accurate to say:

$$\forall x \in \mathcal{D}\ \ (C(x) \wedge L(x))$$

No, it wouldn't be. This statement says every one is a computer science major, and everyone loves logic. We want a statement which says that computer science majors, in particular (not *everyone*), love logic. What we need is an implication:

$$\forall x \in \mathcal{D}\ \ (C(x) \to L(x))$$

What this says *if* x is a computer science major, then x loves logic. Interestingly enough, there might be *no one* in the domain who happens to be a computer science major.

5.1 Reducing/Expanding Domains

Whenever you use quantifiers, you need to refer to some *domain of discourse*. A *domain* is just a set. "of discourse" of the one in discussion. Thus, "domain of discourse" means "the set we're currently talking about". Mathematicians, as you can see, like to use fancy terms to describe relatively simple concepts.

It's interesting to see what effect changing the domain has on the statement. For example, suppose we change the domain from the previous example from "the set of all students" to "the set of computer science majors". In that case, when we translate "All computer science majors love logic", we would simply write:

$$\forall x \in \mathcal{D}\ \ (L(x))$$

We've reduce the domain to a subset (a part of) the original domain. In particular, it's been reduced to the subset we're interested in, namely, computer science majors.

Now, suppose the domain, $\mathcal{S}$ stood for "the set of computer sciences majors who are on the dean's list", and we wanted to say "All computer science majors on the dean's list love algorithms". Suppose, you have the predicate $A(x)$ which stands for "x loves algorithms". The translation would look like:

$$\forall x \in \mathcal{S} \ \ (A(x))$$

But let's say you wanted to *expand* the domain to be "the set of all students", rather than the "set of computer science majors on the dean's list". Then, there needs to be a way of specifying the subset of students which represents computer science majors on the dean's list. So, we add two new predicates. $C(x)$ stands for "x' is a computer science major. $D(x)$ stands for "x is on the dean's list".

The translation would then be:

$$\forall x \in \mathcal{S} \ \ ((C(x) \wedge D(x)) \rightarrow A(x))$$

Perhaps you see a pattern. If you expand the domain, then you need predicates to specify exactly the elements of the expanded domain that you're interested in. You put the predicates on the left-hand side of the implication (on the "hypothesis" side) and the conclusion (which is on the right-hand side) is what you want to say about group of people (or objects).

Let's look at another example. Suppose you wanted to translated "Everyone who is a computer science major or who is on the dean's list loves logic". How do you specify the people in $\mathcal{D}$ that you're interested in? You can write this as: $C(x) \vee D(x)$. What do you want to say about them? That they love logic. So $C(x) \vee D(x)$ is the hypothesis, while $A(x)$ is the conclusion.

$$\forall x \in \mathcal{S} \ \ ((C(x) \vee D(x)) \rightarrow A(x))$$

This leads us to a general way of describing some property of a subset of the domain.

> **General Principle** *Suppose you have a domain $\mathcal{D}$ and you wish to say that everyone in a particular subset of the domain has some property $P(x)$. Then, you need predicates to describe the part of the domain that you wish to talk about. If $A(x)$ describes the part of the domain you wish to say has property $P(x)$, then make $A(x)$ the hypothesis and $A(x)$ the conclusion, i.e., write:*
>
> $$\forall x \in \mathcal{D} \ \ (A(x) \rightarrow P(x))$$

5.2 What Does a Quantifier Really Mean?

When someone says $\forall x \in \mathcal{D} \ \ P(x)$, what are they really saying? If the domain is finite, the semantics (i.e., meaning) is pretty clear. Suppose the domain $D = \{x_1, x_2, x_3\}$. Remember that $\forall x \in \mathcal{D} \ \ P(x)$ is read as "for all x in the domain $\mathcal{D}$, x has property P". "For all" means every single element of the domain, so we have something like:

$$P(x_1) \quad P(x_2) \quad P(x_3)$$

The question is, how should you connect the three parts? Since we are saying property P holds for *every* element, it seems to make sense to connect it with $\wedge$, and in fact, that's correct. So, $\forall x \in \mathcal{D} \ \ (P(x))$, really means:

$$P(x_1) \wedge P(x_2) \wedge P(x_3)$$

at least for the specific domain, $\mathcal{D}$. But you can see how you might apply this same principle for other domains, and even for infinite domains (just an infinite number of $\wedge$ over every element of the domain).

Replacing a $\forall$ quantifier by substituting all the elements of the domain for the variable, and connecting them with $\wedge$ is one kind of *quantifier expansion*. I will use the term quantifier expansion to mean exactly this:

Similarly, when you say $\exists x \in \mathcal{D}\ \ (P(x))$, you are saying that property P is true for *at least one* element of the domain. Instead of connecting the various parts with $\wedge$, we connect them with $\vee$. This is a quantifier expansion of $\exists$. So, $\exists x \in \mathcal{D}\ \ (P(x))$, really means:

$$P(x_1) \vee P(x_2) \vee P(x_3)$$

> **Useful Tip** *Whenever you're wondering how you should interpret statements of the form $\exists x \in \mathcal{D}\ \ (P(x))$ or $\forall x \in \mathcal{D}\ \ (P(x))$, use quantifier expansion to expand the statements using small domains, respectively. This should help you, especially when trying to determine when two statements are equivalent.*

5.3 Scope of a Quantifier

You may have noticed that I write a lot of parentheses when writing quantified expressions. The reason I do this is because you have to be careful when writing expressions. For example, suppose I write:

$$\forall x \in \mathcal{D}\ \ A(x) \vee P(x)$$

Does the $\forall x$ refer to $A(x)$ or to $A(x) \vee P(x)$? By adding parentheses, you make it clear what the quantifier applies to:

$$\forall x \in \mathcal{D}\ \ (A(x) \vee P(x))$$

In particular, the quantifier, $\forall x$ applies to $(A(x) \vee P(x))$. Thus, $(A(x) \vee P(x))$ is said to be within the *scope* of the quantifier $\forall x$. This is very much like scopes in programming languages. When you write a C function and declare local variables, the statements within this function are within the scope of that declaration and any variables referred to in the statements of the function are the variables declared at the beginning of the function.

You might wonder why scope matters. You might write a sentence like:

$$\exists x \in \mathcal{D}\ \ [A(x)] \wedge \exists x \in \mathcal{D}\ \ [P(x)])$$

There are two separate $\exists x$. The first occurrence only has scope over $A(x)$. Thus, we are saying there is some x for which $A(x)$ is true. The second occurrence of $\exists x$ has scope over $P(x)$. Even though both parts say $\exists x$, the x is not necessarily the same in both parts. Thus, the x that is makes $A(x)$ true, may be a different x that makes $P(x)$ true.

Again, just think back to your programming experience. You have two different functions. Each function has x declared as a local variable. Does the x in one function have anything at all to do with the x in the other function? Do they have to have the same value because they are both called x? Of course not. And the same applies here. Each time we have a separate $\exists x$, it has its own scope, and is different from other $\exists x$ (or $\forall x$) that may appear in the statement.

Because this confuses some students of logic, some prefer to change the quantified variables, and we'll discuss when you can and can't change the name of a quantified variable.

5.4 Multiple Quantifiers

It doesn't usually take very long to get used to the idea of a single quantifier, but put two quantifiers together, and this seems to create all kinds of problems. If you're careful, however, you should be able to follow the differences with multiple quantifiers (which usually means, two quantifiers).

Let's pick a predicate to talk about. So far, we've been using one-place predicates, that is, predicates with a single argument. To talk about double quantifiers, we'll use the following two-place predicate:

$P(x, y)$ means "x has earned more credits that y"

Let $\mathcal{D}$ be the "set of all students". What does the following mean?

$$\exists x \in \mathcal{D}\ \ (\exists y \in \mathcal{D}\ \ (P(x, y)))$$

If you read left to right, you'd say "There exists (at least one) x in the domain and there exists at least one y in the domain such that x has earned more credits than y". This basically means some student has earner more credits than some student. Even though x and y are different variables, they could, in principle, refer to the same student (although in this case, it would be difficult for a student to earn more credits than the same student).

Read the statement a little more carefully. "There exists (at least one) x in the domain" (pick an x from the domain) and "there exists (at least one) x in the domain" (pick an y from the domain, which might be x, but doesn't have to be), such that x has earned more credits than y. Notice that when it says "there exists", you should be thinking of picking a particular x and then a particular y. Obviously, since you don't know what's in the domain, you won't really be able to pick a specific x, but pretend that you are able to. This notion of picking elements from the set will become important in a few moments.

Here's an interesting question: is $\exists x \in \mathcal{D}\ \ (\exists y \in \mathcal{D}\ \ (P(x, y)))$ logically equivalent to $\exists y \in \mathcal{D}\ \ (\exists x \in \mathcal{D}\ \ (P(x, y)))$. I've switched the two quantifiers. In the second statement, $\exists y$ appears before $\exists x$. However, the predicate in both statements is still $P(x, y)$.

If you're asking "what does it mean when two quantified statements are logically equivalent"? It means that for all choices of x and y, the two are either true at the same time or false at the same time. So, imagine we find an x and y such that $P(x, y)$ is true. Then, it seems to make sense that both $\exists x \in \mathcal{D}\ \ (\exists y \in \mathcal{D}\ \ (P(x, y)))$ and $\exists y \in \mathcal{D}\ \ (\exists x \in \mathcal{D}\ \ (P(x, y)))$ ought to be true, for exactly the same choices of x and y. That x was picked first and y was picked second in one case, and in the other case y was picked first, then x, doesn't seem to affect the outcome of the either statement. That's correct. In fact, the two statements are logically equivalent.

$$\exists x \in \mathcal{D} \ \ (\exists y \in \mathcal{D} \ \ (P(x,y))) \ \ \equiv \ \ \exists y \in \mathcal{D} \ \ (\exists x \in \mathcal{D} \ \ (P(x,y)))$$

It helps to "expand" the $\exists$ using $\vee$ as we did earlier on. Let's pick a very small domain. Let $D = \{c_1, c_2\}$. Then, the first statement $\exists x \in \mathcal{D} \ \ (\exists y \in \mathcal{D} \ \ (P(x,y)))$ can be expanded by removing the $\exists x$ quantifier to:

$$\exists y \in D \ \ (P(c_1,y)) \vee \exists y \in D \ \ (P(c_2,y))$$

Then, expand each of the $\exists y$ to get:

$$(P(c_1,c_1) \vee P(c_1,c_2)) \vee (P(c_2,c_1) \vee P(c_2,c_2))$$

We do the same with $\exists y \in \mathcal{D} \ \ (\exists x \in \mathcal{D} \ \ (P(x,y)))$ by expanding $\exists y$ first, to get:

$$\exists y \in D \ \ (P(x,c_1)) \vee \exists y \in D \ \ (P(x,c_2))$$

then, expanding $\exists x$ to get:

$$(P(c_1,c_1) \vee P(c_2,c_1)) \vee (P(c_1,c_2) \vee P(c_2,c_2))$$

Because $\vee$ is associative and commutative, you can see the two expanded expressions are logically equivalent to each other. You can apply this idea to any finite domain, and ought to be able to see that it applies to infinite domains as well.

Similarly, using the same idea, you can show that:

$$\forall x \in \mathcal{D} \ \ (\forall y \in \mathcal{D} \ \ (P(x,y))) \ \ \equiv \ \ \forall y \in \mathcal{D} \ \ (\forall x \in \mathcal{D} \ \ (P(x,y)))$$

Having two $\forall$'s and swapping the order of the quantifiers doesn't make a difference.

In both cases, you are saying "every student x in the domain has earned more credits than every student y in the domain". Notice that x and y will, at times, be identical. The way I like to think of double $\forall$ quantifiers is to think of a nested for-loop. Basically, you compare every element in the domain with every element of the domain, to create all possible pairs and say that all possible pairs have the property P.

5.5 $\forall$ and $\exists$ combined

The tricky part of double quantifiers is using one $\forall$ quantifier and one $\exists$ quantifier, and seeing what effect order has on it. To illustrate this idea, I will use two examples. The first example will involve two domains. One domain $\mathcal{P}$ will be the set of problems. The other domain $\mathcal{S}$ will be the set of solutions. We will have a predicate called `solves`(s,p) which means “Solution s solves problem p”.

Can you translate the following statement?

$$\exists s \in S \ \ (\forall p \in P \ \ (\texttt{solves}(s,p))$$

This is not too hard to understand if you read it aloud: “There exists at least one solution, s, such that for all problems, p, s solves problem p”. Or put more naturally, “There is at least one solution that solves all problems”. Because $\exists$ appears first, then you must pick the solution first, and whatever solution is picked, must solve all problems. Note that one can write statements in predicate logic with different domains (although we won't do it very often).

To understand this in more detail, we will do quantifier expansion on $\exists$, by substituting s where the domain $\mathcal{S}$ is $\{s_1, s_2, s_3\}$.

$$(\forall p \in P \ \ (\texttt{solves}(s_1,p)) \ \ \vee \ \ (\forall p \in P \ \ (\texttt{solves}(s_2,p)) \ \ \vee \ \ (\forall p \in P \ \ (\texttt{solves}(s_3,p))$$

What does this say? It says that either s_1 solves all problems in P or s_2 does or s_3 does (recall that all three could solve it—there's nothing about $\exists$ which forces exactly one solution). As a side note, a solution that is supposed to solve all problems is called a *panacea*. Interestingly enough, *panacea* is almost always used in a sentence that goes “such-and-such isn't a panacea” (which means that while some, say, medicine, or plan, might be great, it won't fix everything).

What happens if the order of the quantifiers is switched (but the predicate remains the same)?

$$\forall p \in P \ \ (\exists s \in S \ \ (\texttt{solves}(s,p))$$

Again, read this aloud: “For all problems p, there exists at least one solution s which solves that problem”. Or translated more naturally, “Every problem has some solution”. It doesn't mean every problem must have the *same* solution, just that *some* (possibly different) solution exists for every problem.

Again, quantifier expansion helps us understand what's going on. This time, let $\mathcal{P} = \{p_1, p_2, p_3\}$.

$$(\exists s \in S \ \ (\texttt{solves}(s,p_1)) \ \ \wedge \ \ (\exists s \in S \ \ (\texttt{solves}(s,p_2)) \ \ \wedge \ \ (\exists s \in S \ \ (\texttt{solves}(s,p_3))$$

It says that there exists some solution that solves p_1 *and* there exists some solution that solves p_2 *and* some solution that solves p_3. Even though s appears three times, each has a different scope, so this means that one can pick different solutions for each problem.

It's usually easier to remember that when $\exists$ appears BEFORE $\forall$ then you pick that one individual, and it applies to all the rest. But when $\forall$ comes first, then you get to pick the individual in $\exists$ on a case-by-case basis.

Another example that sometimes helps people remember the distinction between the two uses P, the set of all people and `loves`(x.y) to mean x loves y. So,

$$\exists x \in P \ \ (\forall y \in P \ \ (\texttt{loves}(x, y))$$

gets translated to "At least one person loves everyone".

While

$$\forall y \in P \ \ (\exists x \in P \ \ (\texttt{loves}(x, y))$$

gets translated to "Everyone loves someone", meaning that each person loves some individual but that individual can differ from person to person.

We can summarize the logical equivalences (or lack thereof) of switching quantifiers:

$$\begin{array}{rcl}
\exists x \in D \ \ (\exists y \in D \ \ (P(x, y)) & \equiv & \exists y \in D \ \ (\exists x \in D \ \ (P(x, y)) \\
\forall x \in D \ \ (\forall y \in D \ \ (P(x, y)) & \equiv & \forall y \in D \ \ (\forall x \in D \ \ (P(x, y)) \\
\exists x \in D \ \ (\forall y \in D \ \ (P(x, y)) & \not\equiv & \forall y \in D \ \ (\exists x \in D \ \ (P(x, y))
\end{array}$$

5.6 Change of Quantified Variables

In calculus, you are taught that:

$$\int_1^{10} x^2 \, dx = \int_1^{10} y^2 \, dy$$

Whether you pick x or y doesn't matter. In calculus, this is called a *dummy variable*, meaning that we need to give the variable *some* name, but whether it's x or y doesn't particularly matter–it's the same calculation either way.

Similarly, when you use a quantifier, you can often change variables. For example,

$$\forall x \in D \, (\, P(x) \,)$$

has the same meaning as

$$\forall y \in D \, (\, P(y) \,)$$

In both cases, we are saying "for every x (or y) in the domain". The variable name is needed to it can be referred to, but the choice of x or y is not important. Any variable name will do (at least, in this context).

5.6.1 Bound and Free Variables

Does that mean that you can simply change variables whenever you want? No. But in order to understand when you can and can't change variables, and to understand that, you need to know the difference between a bound and a free variable.

Look at the following formula:

$$\forall x \in D \Big(P(x, y) \to C(x) \Big)$$

The large parentheses represent the *scope* where the quantifier, $\forall x \in D$ apply (namely, it applies to $P(x, y) \to C(x)$). Since the quantified variable is x, then every occurrence of x inside the outer parentheses is bound to the x in the quantifier. Thus, the x in $P(x, y)$ and the x in $C(x)$ are bound to the quantifier x in $\forall x \in D$. On the other hand, y is a free variable, because it does not fall within the scope of a quantifier using the variable y (the quantifier variable is x).

In this example, however,

$$\forall x \in D \Bigg(\forall y \in D \Big(P(x, y) \to C(x) \Big) \Bigg)$$

there are two scopes, one nested inside the other. The scope of $\forall x$ is $\forall y \in D\ (P(x, y) \to C(x))$ (which is the outer scope). The scope for $\forall y$ is $P(x, y) \to C(x)$ which is the inner scope. Since both $P(x, y)$ and $C(x)$ is in the inner scope, the variables fall under the scope of both quantified variables, x and y, thus both x and y are bound.

It's possible for a variable to be bound and free. For example,

$$\forall x \in D \Big(P(x, y) \Big) \to C(x)$$

In this case, the x that appears in $P(x, y)$ is bound since it falls in the scope of $\forall x$. However, the x in $C(x)$ is not bound because it doesn't fall within the scope of a quantified variable x. y is still free since there is no quantified variables.

Finally, it's possible for the same variable to be bound, but bound to different quantifiers.

$$\forall x \in D \Big(P(x) \Big) \to \forall x \in D \Big(C(x) \Big)$$

For example, the x in $P(x)$ is bound to the first $\forall x$ while the x in $C(x)$ is bound to the second $\forall x$. There are two separate $\forall x$ and they are independent of one another. In fact, the statement shown above is exactly equivalent to:

$$\forall x \in D \Big(P(x) \Big) \to \forall y \in D \Big(C(y) \Big)$$

We'll see why momentarily.

At this point, you should be able to tell whether a variable appearing in a predicate is bound or free, and if it's bound, which quantified variable it is bound to. Now we're ready to talk about change of variables.

Change of Quantified Variables *You can change a quantified variable, and all the variables of the same name within the scope to any new variable, as long as this variable does not appear anywhere within the statement as either a free or bound variable within the scope of the quantifier*

Let's go back to the original example.

$$\forall x \in D \Big(P(x,y) \to C(x) \Big)$$

There's only one quantified variable, namely, x. You can change that variable to anything else, as long as it isn't another bound or free variable. For example, we can change x to z since z does not appear within the scope of the quantifier.

$$\forall z \in D \Big(P(z,y) \to C(z) \Big)$$

Notice all the bound variables are changed from x to z. Since the variable was quantified, it really didn't matter whether the variable was called x or z. After all, we are just changing the translation from "for *all* x in the domain" to "for *all* z in the domain". In either case, we're referring to all elements of the domain. Whether x or z is used shouldn't affect the meaning.

Notice that you can't change the bound variable to y, otherwise you'd get:

$$\forall y \in D \Big(P(y,y) \to C(y) \Big)$$

This would not create a statement with exactly the same meaning. In particular, look at $P(y,y)$. The second y, which used to be free, is now bound. Let's see how that affects the meaning of the statement. Suppose $P(x,y)$ means "x swims faster than y" and $C(x)$ means "x gets a prize". Then the statement below,

$$\forall z \in D \Big(P(z,y) \to C(z) \Big),$$

translates to "for all z, if z swims faster than y, then z gets a prize". However, if we had changed the quantified variable to:

$$\forall y \in D \Big(P(y,y) \to C(y) \Big)$$

then, this statement would translate to "for all y, if y swims faster than y, y gets a prize". $P(y,y)$ now has a different meaning and refers to someone swimming faster than themselves, as opposed to $P(z,y)$ which allows others to swim faster than y. This is why you aren't allowed to change a quantified variable to a variable which appears free within the scope of the quantifier.

Can you change a bound variable to another bound variable of the same name is it appears within the scope? No.

$$\exists x \in D \Bigg(\forall y \in D \Big(P(x,y) \Big) \Bigg)$$

Suppose you wanted to change the inner y to an x. Then, you'd get

$$\exists x \in D \Bigg(\forall x \in D \Big(P(x,x) \Big) \Bigg)$$

Again, where you once had $P(x, y)$ where x and y could possibly (and very likely) be different, you now have $P(x, x)$ and compare x to itself. Furthermore, it doesn't seem to make much sense to talk about $\exists x$ and $\forall x$ at the same time. One might argue, rather correctly, that the above isn't semantically valid. To be semantically valid, nested quantifiers must have different variable names.

What happens if the variable isn't within the scope? Then, it's perfectly fine to change the variable. For example,

$$\exists x \in D \Big(P(x)\Big) \land \exists y \in D \Big(Q(y)\Big)$$

Are you allowed to change the y in the second conjunct to x? Yes. That's because x does not appear in the statement within the scope of the quantifier (i.e., x doesn't appear in $Q(y)$). You can rewrite the above equivalently to:

$$\exists x \in D \Big(P(x)\Big) \land \exists x \in D \Big(Q(x)\Big)$$

Just because you changed the y to x doesn't mean that the x in the first part of the statement (i.e., $\exists x \in D \Big(P(x)\Big)$) is related to the x in the second part (i.e., $\exists x \in D \Big(Q(x)\Big)$). Those two x's are different and should be considered different. (Just like if you wrote two function in C, and had two local variables, x. The x in one function has nothing to do with the x in the other function).

You are permitted to change quantified variables to a variable that appears free outside the scope of the quantifier. For example,

$$\exists x \in D \Big(P(x)\Big) \land Q(y)$$

In the above statement, y appears free, but it is outside the bounds of the quantifier. Therefore, you can change the quantified variable to y, and get a logically equivalent statement.

$$\exists y \in D \Big(P(y)\Big) \land Q(y)$$

The y in $Q(y)$ is still free and has not been captured by the $\exists y$ quantifier.

Why change quantified variables? In principle, you never have to. There are two occasions that people change quantified variables. The first is simply to avoid confusion. For example, even though

$$\exists x \in D \Big(P(x)\Big) \land \exists x \in D \Big(Q(x)\Big)$$

is perfectly valid, one might choose to rewrite it as:

$$\exists x \in D \Big(P(x)\Big) \land \exists y \in D \Big(Q(y)\Big)$$

just so a person won't think the x in $Q(x)$ is the same as the x in $P(x)$. Obviously, a person who knows logic well shouldn't have that problem, but sometimes it's better to change variables to avoid problems.

The second time comes when you want to prove a statement. For example, suppose you are asked to prove $\forall x\ P(x)$. However, you managed to prove $\forall y\ P(y)$. Fortunately, $\forall y\ P(y)$ is logically equivalent to $\forall x\ P(x)$ by change of quantified variables, and there's nothing more to prove. Other than that, the issue of changing variables isn't something that comes up all that often (despite the number of pages devoted to explaining it).

6 Statement vs. Formula

What's the difference between a statement and a formula? In a statement, you can say whether it is true or not (provided you are told what the domain is, and what the predicates actually mean). On the other hand, a formula may or may not have a truth value.

This may sound complicated, but in fact, it's rather easy. A statement has no free variables. A formula can have free variables. Since a statement has no free variables, then you can tell whether it's true or false, once the domain and the semantics of the predicate are known (obviously, without that, you can't tell).

For example,

$$\exists x \in D\Big(P(x)\Big)$$

Once you know the domain (say, the set of integers) and what the predicate means (say "x is even), you should, in principle, know whether the statement is true or not (in this case, false). The same statement can be both true for one domain and interpretation of the predicate, but false for another domain and interpretation of the predicate. Nevertheless, we can decide whether it is true or false, once the domain and predicate are picked.

However, the following is a formula, not a statement (whereas all statements can be said to be formulas).

$$\exists x \in D\Big(G(x, y)\Big)$$

While x is quantified, y is a variable, and until we know what y's value is, we don't, in general, know whether this is true or false. y must be known.

Why is it important to talk about statements and formulas? That's because any proof we do will consist of statements and conclusions which are statements. We can't really prove anything about formulas since the free variables don't allow us to conclude whether the formula is true or false.

7 When Is a Double Quantified True?

You're looking at a quantified statement, and trying to decide, is it true for a domain or is it not, you can apply a very useful idea. The idea is to play a "game". If the variable is existentially quantified, you get to pick a value for the variable. If the variable is universally quantified, an opponent gets to pick the value. You are trying to make the statement true, while your opponent tries to make it false. The statement is true if you succeed and false if your opponent succeeds.

Let's look at an example:

$$\exists x \in \mathbb{Z}\ (\forall y \in \mathbb{Z}\ (x \geq y))$$

The expression should be read left-to-right. Since the first quantifier is an $\exists$, you get to pick. The goal is to make $x \geq y$ true. So, naturally, you'd want to pick the largest integer you can. Unfortunately, when picking an integer, you're not allowed to pick infinity. Any integer you pick, must, in principle, be writable, meaning it must consist of a finite number of digits, even though you can use as many digits as you want.

No matter what number you pick, the next part is picked by the "adversary" and that adversary can always pick a number that's one larger than you. So, even if you pick one hundred million, the adversary can pick one hundred million one. Thus, this statement is false for integers.

Let's see what happens when the domain is changed.

$$\exists x \in \mathbb{Z}^-\ (\forall y \in \mathbb{Z}^-\ (x \geq y))$$

In this case, the domain is the negative integers. Again, you pick first since the $\exists$ appears first. Again, the goal is to pick some x such that $x \geq y$. Thus, you want to pick the largest negative integer you can, which happens to be -1. In this case, the adversary can't find a way to pick a number larger than -1, and thus the statement is true.

If the quantifiers are reversed as in:

$$\forall x \in \mathbb{Z}\ (\exists y \in \mathbb{Z}\ (x \geq y))$$

In this case, the adversary begins, with the goal of picking an x such that $x \geq y$ is false. However, no matter what integer the adversary picks, you can always pick a y which is smaller (by subtracting 1).

How about if both quantifiers are $\exists$?

$$\exists x \in \mathbb{Z}\ (\exists y \in \mathbb{Z}\ (x \geq y))$$

In this case, you can choose both x and y. That's not too bad. Pick x to be, say, 3 and y to be 0.

And if both quantifiers are $\forall$?

$$\forall x \in \mathbb{Z}\ (\forall y \in \mathbb{Z}\ (x \geq y))$$

In this case, the adversary chooses both x and y. When that happens, it's difficult to "win". The adversary can pick x to be, say, 0 and y to be 3 and make the statement false.

7.1 Why Does it Work?

Why does this adversary method work? When you are told $\exists x$, you are basically saying, somewhere there exists at least one x, so you can try as hard as you can to find that one x.

For example, if you say "There exists someone who can speak both French and Chinese", then all you have to do is produce one person who can speak both.

On the other hand, when you have a statement that says $\forall x$, then you are saying it has to be true for *all* x in the domain. It has to work for every element. So, you if you let the adversary pick, they will try to find just one element where it fails. For example, when someone says $\forall x\ P(x)$, then for this statement to be false, its negation has to be true. That is, $\sim \forall x\ P(x)$ must be true. This is logically equivalent (using negation of quantifiers) to $\exists x\ (\sim P(x))$, which means you only have to find one x where $P(x)$ is false, and the whole statement is false.

In the adversary game, the adversary attempts to find that one case where the x makes the condition false, and is allowed to pick anything in the domain.

8 Writing Math Predicates

What's a predicate? A predicate is a Boolean function that returns true or false, depending on the value of the arguments. Since quantifiers make statements such as "there exists at least one" and "for all", they can be used to help describe predicates.

In particular, we're interested in giving some math predicates. That is, predicates that appear a lot in mathematics. Let's start by defining two predicates even(x) and odd(x).

$$\begin{aligned}\text{even}(x) &\iff \exists k \in \mathbb{Z}\ [x = 2k]\\ \text{odd}(x) &\iff \exists k \in \mathbb{Z}\ [x = 2k+1]\end{aligned}$$

I use $\iff$ to stand for $\leftrightarrow$ because it's easier to see. It is *not* a new symbol.

You might wonder why even(x) and odd(x) is defined using $\exists$ and not $\forall$. This is a common source of confusion. even(x) is trying to define when a *particular* x in the domain.

Here's an example. Suppose you are trying to determine whether 4 is even. That is, you want to determine even(4). What do you do? Whenever you have a definition, you apply the definition. In this case, you plug in 4 for x in the definition.

$$\text{even}(4) \iff \exists k \in \mathbb{Z}\ [4 = 2k]$$

Thus, 4 is even, if you can find some integer k, such that $4 = 2k$. Does there exists such an integer? Of course. When $k = 2$. Notice that, in fact, there is exactly one integer, k, for any choice of integer x.

We can ask if 3 is even.

$$\text{even}(3) \iff \exists k \in \mathbb{Z}\ [3 = 2k]$$

There is a value k which 3 is twice the size of, namely, 1.5. However, 1.5 is not an integer. So, even(3) is false.

It's common for students to think the following is the definition for even(x).

$$\text{even}(x) \iff \forall k \in \mathbb{Z}\ [x = 2k]$$

However, the above statement says that x is even if x is 2 times every other integer. Thus, even(4) would mean that 4 is equal $2k$ for every $k \in \mathbb{Z}$ which it isn't. The misunderstanding probably comes from believing even(x) defines the set of all even numbers, when it only says whether a specific integer, x is even or not.

Most of you know that $\forall x \in \mathbb{Z}$ [even$(x) \leftrightarrow \sim$ odd(x)]. That is, a number that is even is not odd. It turns out that, given the logic rules you have so far (and ones to be mentioned later on), it will not be possible to prove this obvious statement.

Why not? Logic only provides a basic framework for deriving new facts such as *modus ponens*, *modus tollens*, *proof by contradiction*. In order to derive mathematical facts from logic, you must include *axioms* (basically, premises) which describe how, say, arithmetic behaves. If you look at the definition of even(x) and odd(x), you will see that they are not negations of one another.

However, if by simply allowing some basic arithmetic (basic definitions of adding, subtracting, multiplying and dividing) in addition to the logic which serves as the proof framework, you should be able to prove that $\forall x \in \mathbb{Z}$ [even$(x) \leftrightarrow$ odd$(x+1)$], which is a more straightforward proof.

8.1 Defining Prime and Composite

You probably learned the "definition" of prime and composite back in high school or before. A prime number was considered any number only divisible by itself and 1, while composite is any number that has more than two factors.

Those definitions don't quite capture everything to say about prime and composites. In particular, primes and composites are defined on *positive integers* (meaning integers *greater* than 0). Secondly, 1 is considered neither prime nor composite. Most of you probably believe that 1 should be prime (since it is divisible by itself and 1, and thus follows the definition), but for whatever reason, 1 is not considered prime. It is also not considered a composite. (Just like 0 is neither considered positive nor negative).

Defining what it means to be prime is more complicated than it appears. Here's a first attempt at defining prime:

$$\text{prime}(n) \iff \exists a, b \in \mathbb{Z}^+ \ [\, n = ab \wedge (a = 1 \vee b = 1) \,] \quad \text{(WRONG!)}$$

This definition says that n is prime if there are two *positive* integers such that $n = ab$. Notice that a and b are variables since they are quantified.

This definition of prime is INCORRECT.

Pick a number that's clearly not prime. Say, 12. Apply the (incorrect) "definition" of prime.

$$\text{prime}(12) \iff \exists a, b \in \mathbb{Z}^+ \ [\, 12 = ab \wedge (a = 1 \vee b = 1) \,]$$

Does there exist two such integers? Using the adversarial model, you get to pick both a and b (positive integers) such that $n = ab$ and either $a = 1$ and $b = 1$. Pick a to be 12 and b to 1.

If you complain, "but what if I had picked a to be 3 and b to be 4—remember, you are trying to make the statement *true*, and certainly if you pick a to be 12 and b to 1 it will be true, and certainly you are allowed to pick it. Here's an analogy. Suppose someone says "There is at least one person who received a score of at least 80 on the exam". Then, you find a person who has a score of 81, and says "I found at least one person who has a score of at least 80, so your statement would be true". But what if I complained "Hey, this person had a 50 on their test, so the statement isn't true", would you not say I'm being silly? This is basically confusing, "there exists at least one" with "for every" (even though it may not appear that way).

Let's look at a second (mostly correct) definition of prime numbers. It, too, has a problem, though it's certainly less problematic than

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

All I've done is to change the $\exists$ to $\forall$. In general, changing quantifiers from $\exists$ to $\forall$ won't solve all problems with defining predicates, however. But in this case, it does.

What does it say? It says that for *any* two positive integers a and b which multiply to n, a must be 1 or b must be 1. Let's see if this new definition works (you should always test it with at least one number, just to see if the definition makes sense). Certainly, if n is prime, the definition works (try it). How about if n is composite? Let's use the same example as before, with $n = 12$.

$$\text{prime}(12) \quad \iff \quad \forall a,\, b \in \mathbb{Z}^+ \;\; [\; 12 = ab \rightarrow (a = 1 \vee b = 1) \;]$$

It's true that you can pick $a = 12$ and $b = 1$ and appear to show 12 is prime. However, this time we're using $\forall$, so this means it has to hold true for *any* a and b which multiply to 12. Since we're using the two $\forall$, then if we use the adversary method, the adversary will pick a and b such that the statement above is false. If the adversary picks $a = 2$ and $b = 6$, the statement is false, and thus prime(12) is false.

But where does it fail? Suppose we ask if prime(1) is prime. This definition says that it is. However, 1 is not considered prime. So, we make a small modification to the definition.

The correct definition of prime(n) is:

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

This says that n is prime if and only if $n > 1$ (which now excludes 1 as a choice for prime, and the rest of the previous definition.

You will notice that when using $\forall$, it's common to have $\rightarrow$, and this definition uses it too.

To define composite(n), we write:

The definition of composite(n) is:

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

This definition does use $\exists$ unlike the definition for prime.

You should be able to prove the following (once you know the proof rules):

$$\forall n \in \mathbb{Z}^+ \; [\; \text{prime}(n) \leftrightarrow (n = 1 \vee \text{composite}(n) \;]$$

To prove this, you may need the following (rather obvious) fact about positive integers.

$$\forall n \in \mathbb{Z}^+ \; [\; \sim (n = 1) \rightarrow n > 1 \;]$$

8.2 Defining Rational

Most people "know" the definition of rational is that a number is rational if it's a fraction. However, let's be more precise.

The definition of rational(n) is:

$$\text{rational}(n) \quad \Longleftrightarrow \quad \exists p,\, q \in \mathbb{Z} \;\; [\; n = \frac{p}{q} \wedge q \neq 0 \;]$$

Basically, a number is rational if you can write it as one integer divided by another, and the denominator is not 0 (we don't want to divide by 0). Notice the definition says nothing about whether the fraction is reduced. For example, $\frac{4}{6}$ is rational, even though it could be reduced and written as $\frac{2}{3}$.

Notice that integers are also rational. Any integer n can be written as $\frac{n}{1}$ and satisfy the definition. It turns out irrational numbers are somewhat harder to define. Basically, a number is irrational if it is a real number, but not rational.

8.3 Defining Divides

The definition of "divides" often drives students crazy. It looks very close to division, and yet it isn't division. Here's the definition of divides, which is written using a vertical bar: $|$. Most of the predicates we've defined have a single argument (such as even(n), odd(n), and rational(n)). However, the definition of divides uses two arguments, a and b.

The definition of $a \mid b$ (read it as a divides b) is:

$$a \mid b \quad \Longleftrightarrow \quad \exists k \in \mathbb{Z} \;\; [\; b = a \times k \;]$$

How is this definition different from plain division? In division, / is an operator, and produces a number as a result. For example, you write 12/3 and get 4 as the answer. What do you notice about division? *It's not a predicate!* That's right. It's an operator which produces a number as a result. However, $|$ (read as "divides") is a predicate. Thus, it returns, as a result, either `true` or `false`.

The second part of the definition that seems unusual is that the numbers are reversed. You write $4 \mid 12$ and read "4 divides 12", rather than the more "natural" 12 is divisible by

4. If you ask why it's written in reverse, I won't have a good answer. But the definition is that of divisibility.

Thus, if someone says $4 \mid 12$ you would say "yes, that's true" (since it's a predicate). However, $5 \mid 12$ is false and $12 \mid 4$ is false as well.

If $4 \mid 12$ is indeed true, then you should be able to apply the definition of divides to see if it is indeed true.

$$4 \mid 12 \quad \Longleftrightarrow \quad \exists k \in \mathbb{Z} \ \ [\, 12 = 4 \times k \,]$$

Thus, there exists some constant k which happens to be an integer, which, when multiplied by 4 gives you 12. By some simple algebra, you get $k = 3$. So, yes, $4 \mid 12$ is indeed true.

8.3.1 Defining Predicates—Shortcuts

If you look at the definition of divides, it's hard to tell what domain a and b belong to. When predicates are defined, they should really be defined using universal quantifiers so one can tell what the domains the arguments have, thus, we should write:

$$\forall a, b \in \mathbb{Z}\, [\, a \mid b \iff \exists k \in \mathbb{Z} \ \ [\, b = a \times k \,]\,]$$

However, usually the outer $\forall$ quantifiers aren't used to make it easier to write. In this case, it's useful to write it down to show that a and b are integers.

All the definitions of predicates could have been defined with $\forall$ just to let you know which domain the variable in the predicate refers to, such as:

$$\forall x \in \mathbb{Z}\, [\, \text{even}(x) \quad \Longleftrightarrow \quad \exists k \in \mathbb{Z} \ \ [x = 2k]\,]$$

Again, it's not very common to do so unless you really want to explicitly note out the variables' domain.

8.4 Defining Congruence

While the definition of *divides* is somewhat unusual, at least it's related to divisibility. Congruence is related to the mod operator in C/C++ programming. However, it's not nearly the same. Here's how it's defined:

> *The definition of* $a \equiv b \pmod{n}$ *(read this as "a is congruent to b, mod n" is:*
>
> $$\forall a, b \in \mathbb{Z}\ \forall n \in \mathbb{Z}^+\, [\, a \equiv b \pmod{n} \iff \exists k \in \mathbb{Z}\ (\, a = nk + b\,)\,]$$

You should read the definition carefully. Basically, it says that a and b are any integers and n is a positive integer, and they are congruent $(\text{mod } n)$ if and only if there is some integer k, such that $a = nk + b$.

This seems like an unusual definition, and it may not even make sense. The intuition behind this is that a is congruent to b $(\text{mod } n)$ when the remainder of dividing a by n and the remainder of dividing b by n is the same. The definition doesn't appear to say this, but I'll demonstrate in a moment that it really does say that.

But before we do that, let's define what is meant by a *remainder*.

8.4.1 The Quotient-Remainder Theorem

Here's the statement of the quotient-remainder theorem.

$$\forall n \in \mathbb{Z} \left(\forall d \in \mathbb{Z}^+ \; (\; \exists! \; q, r \in \mathbb{Z} \; \; [\; n = dq + r \; \wedge \; 0 \le r \le d - 1 \;] \;) \right)$$

This is hard to read, but basically it says that if you take any integer n and divide it by a positive integer d (the divisor), you will have a unique quotient q (the result of the division) and remainder r as long as r is between 0 and $d - 1$, inclusive. The $\exists!$ means "there exists exactly one", which will be defined in detail later on.

This theorem basically says that you have a unique quotient and remainder given any integer n and any positive integer d.

I won't go through this proof, since I'm only interested in discussing what the remainder means. To get a better understanding, let's apply the quotient-remainder theorem. Suppose n is 17 and d is 7, can we find values for q and r such that:

$$17 = 7q + r \; \wedge \; 0 \le r \le d - 1$$

What we're basically doing is dividing 17 by 7 to get a quotient and a remainder. This division would give us 2, with a remainder of 3. Thus, $q = 2$ and $r = 3$. Notice that these values of q and r are unique. You can't find another pair of values which satisfy the formula above. Notice there is a *second* constraint, which says that the remainder must be between 0 and $d - 1$ inclusive.

This constraint is not that surprising. When you divide an integer by 7, what do you expect the remainder to be? The choices are 0, 1, 2, 3, 4, 5, 6. Thus, the remainder r is constrained to $0 \le r \le 6$. Since we're dividing by 7, then the divisor is d is 7. Thus, the remainder is constrained by: $0 \le r \le d - 1$. This makes the definition of remainder reasonable, doesn't it?

However, because the remainder is never negative, it does yield unusual results when n is a negative number. For example, suppose n is -17, and d is 7.

$$-17 = 7q + r \; \wedge \; 0 \le r \le d - 1$$

What is q and r? You'd normally do a back-of-the-envelope calculation and divide -17 by 7 to get -2 with some remainder. However, that turns out to be incorrect.

The answer is really $q = -3$ and $r = 4$. Why is q equal to -3? Because r is non-negative, it always adds something to dq, thus, q has to be picked so that it is "more negative" than n. That is, a must be picked so that $dq < n$ which means it is usually $n/d - 1$ where n/d is defined to be n divided by d with the remainder thrown away. Thus, $-17/7$ would be -2, and then subtract 1 to get -3. The quotient-remainder theorem is therefore less intuitive when n is negative.

Notice that this definition of quotient and remainder is usually not the same as the result of integer division and the mod operator in a language like C or C++. They define / and % in different ways. In particular, it's possibly for the remainder to be negative in these languages (and for the divisor, d to be negative too).

These are all side issues. The main point I want to make is that a remainder likes between 0 and $d - 1$. This may have unusual consequences, but it's a reasonable definition.

8.4.2 Comparing Remainders

When dividing 17 by 7, the remainder was 3. What's the next larger number to have a remainder of 3? The answer turns out to be 24. Just add 7 to 17. That makes sense. After all, when you divide, you will simply have a quotient which is 1 larger than before. The next larger number to have a remainder of 3 (when dividing by 7) is 31.

Here's a list of the numbers with remainders of 3, when dividing by 7.

$$\{\ldots, -4, 3, 10, 17, 24, 31\ldots\}$$

All I've done is to add or subtract 7 repeatedly starting with the number 17.

Do you believe that -4 has a remainder of 3? Let's apply the definition. Plug in $n = -4$, $d = 7$, and $r = 3$.

$$-4 = 7q + 3$$

Solving for q gives us $q = -1$.

In fact, we can describe all the values which have the same remainder as 17 divided by 7 using a truth set as:

$$\{x \in \mathbb{Z} \mid \exists k \in \mathbb{Z}\ [\ 17 + 7k\ ]\}$$

This set is the set of all numbers x which is generated by adding multiples of 7 (the multiple can be negative!).

This idea applies to any integer n and any positive divisor d. You can find all numbers with the same remainder as n divided by d by adding positive or negative multiples of d to n.

Thus, two numbers, a and b, have the same remainders (when dividing by d) when you can add a multiply of d to b to get a. And, guess what? That's the definition of congruence!

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

In particular, pay attention to $a = nk + b$. In congruence, you write: $a \equiv b \pmod{n}$. where to say that a and b have the same remainders when dividing by n. Notice that nk refers to the multiples of n that you use to add to b to get a.

8.4.3 Useful Fact

8.4.4 Alternate Definition of Even

8.5 Definitions—Memorize Them! Apply Them!

So, far we've defined the following predicates:

- even(x)
- odd(x)
- prime(n)
- composite(n)
- rational(x)
- $a \mid b$ (read as: "a divides b")
- $a \equiv b \pmod{n}$ (read as: "a is congruent to b (mod n)").
- $\lfloor x \rfloor$ (read as: "floor of x"—note: defined in text)
- $\lceil x \rceil$ (read as: "ceiling of x"—note: defined in text)

There are 9 predicates. You should know memorize all of them (especially the last 4). Why? Whenever you do proofs with these predicates, invariably, you will need to know the definition.

It's common error is to use some sort of intuitive definition. For example, you might be asked to prove $p \mid r$ (that is, p divides r. If you don't know the definition, then you're either going to have to look it up (and surprisingly, many people don't) or you will be "using the force". That is, you will be using an intuitive definition which may have nothing to do with the real definition.

As much as I keep repeating the importance of knowing the definitions, time and again, students absolutely refuse to learn the definitions. In some ways, I can't blame most students. You tell 100 students to do something, 10 students will do it, 20 more will say they thought about doing it, but never got around to it, 30 more will vaguely recall you said to do something, and 50 more will swear you never said it at all. Which part of the class are you?

9 Exactly One, At Least Two

While the quantifiers $\exists$ and $\forall$ allow you to say "at least one" and "for all", it's not nearly so easy to say "exactly one" or "at least two". "at least one" is not the same as "exactly one". I used the modified quantifier $\exists!$ to stand for "there exists exactly one" because the notation was simpler to use. In a moment, you're going to see why it's difficult.

To say there is exactly one x which satisfies some predicate P, you must show that no one else but x satisfies it. How do you do this? This is how:

$$\exists x \in \mathcal{D}\ \Big(P(x) \to \forall y \in \mathcal{D}\ [\ P(y) \to x = y\ ] \Big)$$

How can this be translated to English (or at least math-like English)? It says there exists at least one x where $P(x)$ is true, and for every y in the domain (which also includes x), if $P(y)$ is true, then y must be x. For example, if the predicate $P(x)$ means "x won a gold medal", then the above says that there exists at least one person who won the gold medal, and for every y who won a medal, they must be x.

As you can see, in order to say "exactly one", you need to to have equality defined (i.e., =), or at least, it's very convenient to have equality defined. In logic, equality is very powerful, and allows you to say such things as "exactly one" in a somewhat convenient way.

If you don't like the definition, you might use the contrapositive of the implication to say:

$$\exists x \in \mathcal{D} \left(P(x) \rightarrow \forall y \in \mathcal{D} \, [\, x \neq y \rightarrow \sim P(x) \,] \right)$$

This says that for every y who is not x, then $P(y)$ must be false. Some of you may prefer this definition.

9.0.1 At Least Two

To say at least two, we need two individuals. What's wrong with the following statement? That is, why doesn't it say "at least two individuals have the property P"?

$$\exists x,\, y \in \mathcal{D} \left(P(x) \wedge P(y) \right)$$

Do x and y have to refer to different element of $\mathcal{D}$? The answer is no, they don't. Just because they have different names doesn't mean that x and y have to refer to different elements. In fact, the statement above is exactly the same as:

$$\exists x \in \mathcal{D} \left(P(x) \right)$$

Surprising, isn't it? How can we solve this problem? The answer is to make sure x and y are different individuals and this is done by using the equality operator (more precisely, the inequality operator).

$$\exists x,\, y \in \mathcal{D} \left(P(x) \wedge P(y) \wedge x \neq y \right)$$

Notice that nothing prevents 3 elements of $\mathcal{D}$ (or more) from having the property P.

9.0.2 Exactly Two

To say, exactly two elements of the domain satisfy P, you need to say that at least two element satisfy P, but any third that satisfies P must be one of the first two elements. This is how it's written.

$$\exists x,\, y \in \mathcal{D} \left(P(x) \wedge P(y) \wedge x \neq y \wedge \forall z \in \mathcal{D} \, [\, P(z) \rightarrow z = x \vee z = y \,] \right)$$

Now that you know how this works, you should be able to define at least three and exactly three.

9.0.3 At Most Two

This is actually somewhat more challenging that you might expect, all though not entirely. It turns out that it's not hard to say at most two (but at least one). You take the definition of exactly two, and remove $x \neq y$.

$$\exists x,\, y \in \mathcal{D}\, \Big(P(x) \wedge P(y) \wedge \wedge \forall z \in \mathcal{D}\, [\, P(z) \rightarrow z = x \vee z = y\,] \Big)$$

Basically, this says that x and y satisfy P, but they *could* be the same element. However, a third element z must be the same as either x or y. This means that there's no way to have 3 different elements satisfy P. Thus, the above says, at least one, but at most two.

To really say "at most two", you must allow for the fact that no one satisfies the predicate, P. This is simply written as:

$$\sim \exists x \in \mathcal{D}\, \Big(P(x) \Big)$$

which says that no one exists that satisfies P.

So, combine the above two predicates with a $\vee$ to get:

$$\sim \exists x \in \mathcal{D}\, \Big(P(x) \Big) \vee \exists x,\, y \in \mathcal{D}\, \Big(P(x) \wedge P(y) \wedge \wedge \forall z \in \mathcal{D}\, [\, P(z) \rightarrow z = x \vee z = y\,] \Big)$$

which says "either no one satisfies P, or at least one, but at most two satisfy P" which is the same as saying "at most two satisfy P". Again, complicated, but that's the contortions needed to say something like "at most two".

10 New Logical Equivalences

Finally, we get to the useful part. There is already a large list of logical equivalences for propositional logic. Essentially, all of those equivalences still hold (such as DeMorgan's Laws).

Since predicate logic introduces quantifiers, it's no surprise that these new equivalences involve the use of quantifiers. Basically, there are only three logical equivalences.

Here's the first one:

$$\sim \forall x \in \mathcal{D}\, [\, \alpha\,] \iff \exists x \in \mathcal{D}\, [\, \sim \alpha\,]$$

Suppose α was the formula $P(x)$ where $P(x)$ meant "x received a cookie". Then, what does it mean when someone says "It is not the case that everyone received a cookie". This means there must be at least one person who did not receive a cookie. That's what the above equivalence says.

Similarly,

$$\sim \exists x \in \mathcal{D}\, [\, \alpha\,] \iff \forall x \in \mathcal{D}\, [\, \sim \alpha\,]$$

Using the same predicate as before, this says "It is not the case that at least one person received a cookie" is the equivalent of saying "no one received a cookie" (or to follow the translation exactly, for all x, x did not receive a cookie").

Finally, there's change of quantified variables.

$$\forall x \in \mathcal{D} \ [\ \alpha\ ]$$

can be rewritten as:

$$\forall y \in \mathcal{D} \ [\ \sim \alpha\ ]$$

provided that y does not appear in α.

Similarly,

$$\exists x \in \mathcal{D} \ [\ \alpha\ ]$$

can be rewritten as:

$$\exists y \in \mathcal{D} \ [\ \sim \alpha\ ]$$

provided that y does not appear in α.

Basically, you can change the quantified variable as long as the new variable does not appear in α.

As before, you may apply logical equivalences to parts of statements, but you may NOT apply proof rules to parts of statements.

11 New Proof Rules

There are four new proof rules using quantifiers. You may still use all the old proof rules from propositional calculus (e.g., DeMorgan's Laws, Double Negative, Associativity, Contrapositive, etc).

11.1 Universal Instantiation

This is also called "Universal Elimination". The rule says if you've already proved or have assumed $\forall x\ \alpha(x)$, then you can say $\alpha(a)$ for any constant you want. What am I using $\alpha(x)$ when predicates use uppercase letters (e.g., $P(x)$? That's because you can use any well-formed statement with *free* variable x appearing in it. Thus, $\alpha(x)$ could be as simple as $P(x)$ or more complex as $P(x) \vee Q(x)$. We'll see several examples of applying Universal Instantiation in the next subsection to illustrate.

This pseudo-proof shows how Universal Instantiation is applied.

i	$\forall x\ [\ \alpha(x)\ ]$	
	$\dots$	
j	$\alpha(a)$	$\forall$ I, x = a, step i

Assume that you have already proved step i (or it is assumed), then you can prove step j

using universal instantiation (which is abbreviated $\forall$ I), where x has been instantiated to a. The justification includes the abbreviated name of the proof rule, $\forall$ I (short for Universal Instantiation), followed by $x = a$, to indicate that variable x has been replaced by constant a, and then the step number which this rule as applied to.

In step j, a is a constant. The constant you pick can be new (meaning it hasn't appeared in the proof at all), or it can be previously used. It doesn't matter. After all, the predicate is true for all elements in the domain. Obviously, you want to pick a constant that will help you to do the proof, but the rule allows you to pick whichever constant you want, regardless of why.

The only restriction to this rule is that the domain of discourse must be non-empty. If the domain were empty (i.e., had no elements), you could not apply this rule. After all, how can you instantiate when there's nothing to instantiate to?

11.1.1 Examples of Universal Instantiation

Here's a simple application of Universal Instantiation. $\alpha(x)$ appears as $P(x)$. Assume all steps up to, but not including j have been proved. At step j, you can conclude $P(a)$. Of course, you can conclude that right after step i has been proved.

i	$\forall x$ [$P(x)$]	
	$\ldots$	
j	$P(a)$	$\forall$ I, x = a, step i

You may, of course, pick a different constant, such as b, instead of a. b might even have already appeared earlier in the proof. For example, b appears in step $i - 1$.

i - 1	$\forall x$ [$Q(b)$]	
i	$\forall x$ [$P(x)$]	
	$\ldots$	
j	$P(b)$	$\forall$ I, x = b, step i

$\alpha(x)$ can be more complicated. For example, it might be $P(x) \lor Q(x)$.

i	$\forall x$ [$P(x) \lor Q(x)$]	
	$\ldots$	
j	$P(a) \lor Q(a)$	$\forall$ I, x = a, step i

Notice that all the occurrences of x (which are bound) have been replaced by a. Thus, at step j, you can conclude $P(a) \lor Q(a)$.

You can even have other variables. For example, $\alpha(x)$ could be $\exists y$ [$P(x) \rightarrow Q(x, y)$], which contains both x and y. Notice that this is *still* Universal Instantiation, because $\forall$ is the main operation (it surrounds the entire expression).

$$\begin{array}{r|ll} \text{i} & \forall x\ [\ \exists y\ [\ P(x) \rightarrow Q(x,y)\ ]\ ] & \\ & \ldots & \\ \text{j} & \exists y\ [\ P(a) \rightarrow Q(a,y)\ ] & \qquad \forall\text{ I, x = a, step } i \end{array}$$

All occurrences of x within the scope of the quantifier have been replaced by the constant a. However, y, which was *not* instantiated, was left alone. It will have to be instantiated separately.

Finally, in principle, x doesn't even have to appear in $\alpha(x)$ at all. For example, $\alpha(x)$ could be $\exists y\ [\ P(y) \rightarrow Q(y)\ ]$. Thus, you would get:

$$\begin{array}{r|ll} \text{i} & \forall x\ [\ \exists y\ [\ P(y) \rightarrow Q(y)\ ]\ ] & \\ & \ldots & \\ \text{j} & \exists y\ [\ P(y) \rightarrow Q(y)\ ] & \qquad \forall\text{ I, x = a, step } i \end{array}$$

Since x doesn't appear at all, it doesn't matter which constant is used in the justification in step j.

11.1.2 Common Errors

Perhaps the most common error that occurs when writing proofs is *applying proof rules to part of a statement*. You are permitted to replace a substatement (i.e., part of a statement) with a logically equivalent statement (because the two meanings are identical), but you *aren't* permitted to replace part of a statement with a statement using *proof rules*, at least, not in general. This leads to incorrect facts being proven.

For example, consider the following partial proof.

WRONG PROOF BELOW

$$\begin{array}{r|ll} \text{i} & \forall x\ [\ P(x)\ ] \rightarrow \forall y\ [\ Q(y)\ ] & \\ & \ldots & \\ \text{j} & P(a) \rightarrow \forall y\ [\ Q(y)\ ] & \qquad \forall\text{ I, x = a, step } i \end{array}$$

Why is this incorrect? Look at step i. It says that predicate P has to be true for the entire domain, before you can conclude that predicate Q has to be true for the entire domain. You can't then simplify that to $P(a)$ in step j. Step j is an INCORRECT application of Universal Instantiation. The main connective of step i is $\rightarrow$, NOT the

Here's a specific example to show why it's wrong. Suppose $P(x)$ means "x votes to have the exam next week" and $Q(y)$ means "y will take the exam next week". Then, statement i would translate to "If everyone votes to take the exam next week, then everyone will take the exam next week". Statement j would translate to "If student a votes to have the exam next week, then everyone will take the exam next week". You can't conclude statement j from statement i.

You can only apply Universal Instantiation when the entire statement matches the pattern $\forall x[\ \alpha(x)\ ]$ (where x could be any other variable. If the $\forall$ appears as *part* of a larger statement, as it did in the previous example, Universal Instantiation can't be applied.

11.2 Existential Addition

This rule is also called "Existential Generalization". The rule says if you've already proved or have assumed $P(a)$ for some constant a, then you can say $\exists x\ P(x)$ for any variable you want. Obviously, if P is true for a, then there exists some x that it must be true for (since $P(a)$ is true, the domain has to be non-empty).

i	$P(a)$	
	$\dots$	
j	$\exists x\ [\ P(x)\ ]$	$\exists$ Add, a / x, step i

Thus, if $P(a)$ was proved in step i, then you can conclude $\exists x\ [\ P(x)\ ]$ using existential addition on step i, with a replaced by x.

11.3 Existential Instantiation

This is also called "Existential Elimination". If someone says $\exists x\ P(x)$, then you can pick a constant that represents the element in the domain for which P is true. However, this is not a *known* constant.

Here's an analogy. Suppose you are voting for a class president. The only people eligible for that position is someone in the class and only one person will win. You can say, "Since we're going to have a vote on class president, someone will win. Let's call that person 'b'. If 'b' wins, then 'b' will have to give an inaugural speech, and 'b' will have to attend all Student Council meetings, and 'b' will give a speech during the graduation ceremony". You don't know who 'b' is now, but you can give 'b' a name.

This is how the rule is applied.

i	$\exists x\ [\ P(x)\ ]$	
	$\dots$	
j	$P(a)$	$\exists$ I, x = a, step i

There is a restriction. a must be a completely new constant. It can not have appeared earlier in the proof. Why not? Suppose you did not have this restriction. Then, you could prove nonsense like:

BAD PROOF BELOW

3	$\exists x$ [even(x)]	$\ldots$
4	$\exists x$ [odd(x)]	$\ldots$
5	even(a)	$\exists$ I, x = a, step 3
6	odd(a)	$\exists$ I, x = a, step 3
7	even(a) $\wedge$ odd(a)	$\wedge$ Add, step 5, 6
8	$\exists x$ [even(x) $\wedge$ odd(x)]	$\exists$ Add, step 7

This proof would allow you to conclude that there is some even number (step 5), then conclude that same number is also odd (step 6), which would allow you to claim there is some number that's both even and odd, which is ridiculous. We don't want our logic to prove incorrect facts.

The error is in step 6. You can't reuse the constant that was used in step 5 (or anywhere). In particular, you run into problems when the constant picked is one that was existentially instantiated. However, rather than keep track of how constants were instantiated, just use the much simpler rule of: "When using existential instantiation, always pick a new constant to instantiate".

11.3.1 Can You Use Existential Instantiation Twice?

Does this rule allow you to instantiate twice? The answer turns out to be "yes". For example,

3	$\exists x$ [$P(x)$]	$\ldots$
4	$P(a)$	$\exists$ I, x = a, step 3
5	$P(b)$	$\exists$ I, x = b, step 3

In step 4, the x in $P(x)$ was instantiated to a, a new constant. In step 5, the x in $P(x)$ was instantiated to b, a new constant. Does this mean there are two different elements in the domain that satisfy P? This doesn't seem to make sense. After all, $\exists x$ [$P(x)$] only guarantees that one element of domain satisfies it. How can we claim there are two elements?

The key is that a and b might refer to exactly the same element. There's no guarantee this will happen, of course. All you can say is a and b might be the same element, or it might not be. Just like a person might go by the name "Alex" and "Sasha" and both names refer to the same person ("Sasha" is a common Russian nickname for "Alex"), similarly, different constants in logic can refer to same element of the domain (unless you can show that $a \neq b$, in which case they would have to be different constants).

11.4 Universal Instantiation

This is the only one of the four new proof rules which requires an assumption. This proof rule answers the question: how do we prove something like $\forall x$ [$P(x)$]? It doesn't seem to make sense to find all possible constants, and then prove each of those constants are true.

After all, the domain could be infinite.

Instead, the idea is to assume the existence of a completely new constant. That way, you make no assumptions about that constant. If you were to use a previously used constant, you would run into problems (which we'll see later on). Then, anything you prove about this new constant ought to be true of any constant in the domain. This argument is known as the argument of the "generic particular". You pick an element from the domain by giving it a name (a constant). You prove something about that constant. Since the constant could be any element of the domain, and nothing was previously assumed about it, then it must be true of any element of the domain.

Let's see how this works:

	$\ldots$	
i	$a \in \mathcal{D}$	Assume
	$\ldots$	$\ldots$
j	$P(a)$	$\ldots$
j + 1	$\forall x\ [\ P(x)\ ]$	$\forall$ Add, a/x, step i–j

Given that you've started a subproof at step i and assumed a, a completely new constant in the proof, then eventually proved, at step j, that $P(a)$ as true, then after this subproof, you may conclude, at step $j + 1$, that $\forall x\ [\ P(x)\ ]$ is true using Universal Addition (also called "Universal Generalization") which is written as $\forall$ Add, followed by a/x which means constant "a" has been replaced by variable x, and then the step numbers.

One of the more important parts of this rule is that the constant assumed in step i (at the beginning of the subproof. *must be unused!*. If a, the constant, had previously been used, you would be able to prove incorrect facts. That's why the constant used in the assumption must be completely new.

By the end of the subproof, you will want to prove $P(a)$ (at step j in the above example). That step needs to be justified. Typically, as you write the proof, you will fill in step i, step j, and step $j + 1$. You will then try to figure out what other steps are needed to get from step i to step j.

Once you have proved step j, you can close off the subproof. In step $j + 1$, you can then conclude $\forall x\ [\ P(x)\ ]$. This is the "generalization" step. You've proved something about an arbitrary a from the domain (this is the "generic particular") and conclude $\forall x\ [\ P(x)\ ]$ from that.

So, if someone asks you "How can you prove a statement with a universal quantifier?", you can reply that you must use the "Universal Addition" rule, which requires an assumption of some element in domain, and prove something about that element.

11.4.1 Why a New Constant?

In Universal Addition, you start a subproof and assume a new constant (don't get this confused with Universal Instantiation where you can introduce any constant). Why must you do this?

Consider the following argument:

$$\begin{array}{l} \exists x\,[P(x)\,] \\ \hline \therefore\ \forall x\,[P(x)\,] \end{array}$$

Does this seem like valid argument? For example, if the predicate $P(x)$ meant "x is even", then the argument would say that given there exists some even number, then all numbers would be even. This sounds false, and indeed it is. We don't want our proof rules to be able to prove something like this. Yet, it can, if you don't pick a new constant.

For example,

WRONG PROOF BELOW

$$\begin{array}{r|ll}
3 & \exists x\,[P(x)\,] & \ldots \\
4 & P(a) & \exists\text{ I, x = a, step 3} \\
 & & \\
5 & \quad \boxed{a \in \mathcal{D}} & \text{Assume} \\ \cline{3-3}
6 & \quad \boxed{P(a)} & \ldots \\
7 & \forall x\,[P(x)\,] & \forall\text{ Add, steps 5–6}
\end{array}$$

Where's the error? It occurred in step 5, where a was assumed. In a Universal Addition subproof, the constant must be completely new. Thus, you would have written:

$$\begin{array}{r|ll}
3 & \exists x\,[P(x)\,] & \ldots \\
4 & P(a) & \exists\text{ I, x = a, step 3} \\
 & & \\
5 & \quad \boxed{b \in \mathcal{D}} & \text{Assume} \\ \cline{3-3}
\end{array}$$

At this point, you'd be stuck and be unable to proceed with anything new.

But what if you decided to be clever, and start the assumption right after step 3, and instantiate later.

WRONG PROOF BELOW

$$\begin{array}{r|ll}
3 & \exists x\,[P(x)\,] & \ldots \\
 & & \\
4 & \quad \boxed{a \in \mathcal{D}} & \text{Assume} \\ \cline{3-3}
5 & \quad \boxed{P(a)} & \exists\text{ I, x = a, step 3} \\
6 & \forall x\,[P(x)\,] & \forall\text{ Add, steps 5–6}
\end{array}$$

The error is then in step 5. In Existential Instantiation, you must also pick a brand new constant. Thus, at step 5, you can't pick a, since it already appeared in step 4 as a new constant. You would have to pick another new constant for step 5.

11.4.2 Errors Using Universal Addition

You must be careful and generalize for all values of a. For example, suppose you had the following:

WRONG PROOF BELOW

	$\ldots$	
i	$a \in \mathcal{D}$	Assume
	$\ldots$	$\ldots$
j	$P(a,a) \rightarrow Q(a)$	$\ldots$
j + 1	$\forall x \; [\; P(x,a) \rightarrow Q(x) \;]$	$\forall$ Add, a/x, step i–j

Step $j+1$ has an error. When you generalize, you can't generalize some of the a's that appear in step j. You must generalize on all occurrences of a in the last step of the subproof (i.e., step j).

Why is this an incorrect step to take? Let's let $P(x,y)$ mean "x likes y" and $Q(x)$ mean "x is successful". Then, step j would translate to "If a likes himself or herself, then a will be successful". You would expect the generalization to be $\forall x \; [\; P(x,x) \rightarrow Q(x) \;]$ which says that "anyone who likes themselves will be successful". However, by generalizing on two of the a's, but leaving the other alone, you get the result that everyone who likes a will be successful, which is not at all what you expect to conclude. Thus, you *must* generalize to $\forall x \; [\; P(x,x) \rightarrow Q(x) \;]$ to use $\forall$ Add correctly.

12 Correct Conclusion–Incorrect Justification

The problem with learning logic rules is that, like anything in life, it's easy to skip steps. People do this all the time. Skipping steps makes life easy. You would think that, as computer science majors, we'd all be extremely careful and be extremely precise, always making sure to every last step. After all, this is what programming requires, right?

And yet, it's just human nature not to find ways to make life easier, to quickly skip over the obvious and conclude things we know to be true, even as the rules do not permit it. Here are a few examples.

WRONG PROOF BELOW

3	$\forall x \; [\; \sim P(x) \wedge Q(x) \;]$	$\ldots$
4	$\forall x \; [\; \sim P(x) \;]$	$\wedge$ Simp, step 3

Suppose you've already proven step 3. Is step 4 correct? It turns out, the fact is correct. After all, if predicate P and Q is true for all elements of the domain, you'd expect it to be P to be true of all elements of the domain, too. Nevertheless, it is *not* $\wedge$ simplification. Why not? Because the main connective is *not* $\wedge$. The main symbol is the quantifier $\forall$. The best you can do, at this point, is to apply universal instantiation.

The real proof requires a bit more work. Since the goal is to prove a $\forall$, you will need to use

Universal Addition, which requires a subproof.

3	$\forall x\,[\,\sim P(x) \wedge Q(x)\,]$	...
4	$a \in D$	Assume
5	$\sim P(a) \wedge Q(a)$	$\forall$ I, x = a, step 3
6	$\sim P(a)$	$\wedge$ Simp, step 5
7	$\forall x\,[\,\sim P(x)\,]$	$\forall$ Add, step 3–6

So, it *can* be proved, but it takes 4 steps to do it (and a subproof), instead of one step.

Let's look at another example:

WRONG PROOF BELOW

3	$\forall x\,[\,P(x)\,] \to \forall x\,[\,Q(x)\,]$	...
4	$P(a)$	...
5	$Q(a)$	$\forall$ MP, step 3

Suppose you've proved steps 3 and 4. Can you conclude step 5? Again, you can't. To use universal *modus ponens*, step 3 has to be written as: $\forall x\,[\,P(x) \to Q(x)\,]$, which it isn't.

Not only is the above proof "incorrect" because it uses the wrong rules, it also happens to be incorrect. You can use quantifier expansion to see why. Suppose you had three elements in the domain, and referred to them as a, b, and c. Thus, statement 3 expands out to:

$$P(a) \wedge P(b) \wedge P(c) \to Q(a) \wedge Q(b) \wedge Q(c)$$

Now suppose $P(a)$ is true, but $P(a)$, $P(b)$, $Q(a)$, $Q(b)$, and $Q(c)$ are all false. Then, the above statement is true (since $P(a) \wedge P(b) \wedge P(c) = T \wedge F \wedge F = F$ and $Q(a) \wedge Q(b) \wedge Q(c) = F \wedge F \wedge F = F$, so $F \to F = T$). Step 4 would also thus be true. But step 5 which concludes $Q(a)$ is true is incorrect. This is the kind of faulty reasoning you can get if you fail to use the rules correctly.

Here's an example of an incorrect rule, but leads to a correct result.

WRONG PROOF BELOW

3	$\forall x\,[\,P(x) \to \exists y\,[\,Q(y)\,]$	...
4	$\forall x\,\exists y\,[\,P(x) \to Q(y)\,]$	$\exists$ Add, step 3

Suppose step 3 were proved. Can you conclude step 4? The answer is no, there's no such rule to let you conclude the statement in step 4 from step 3. Stare at step 4, and you might be able to see how it comes from step 3. After all, does it really matter where you write $\exists y$? You have to pick the y, and as long as it's picked after (i.e., to the right of) the $\forall x$, it should be fine.

However, to show step 4 comes from step 3 requires additional work. In particular, you want to prove a universal quantifier in step 4. What do you do? Start a Universal Addition subproof!

3	$\forall x\ [\ P(x) \to \exists y\ [\ Q(y)\ ]$	$\ldots$
4	$a \in D$	Assume
5	$P(a) \to \exists y\ [\ Q(y)\ ]$	∀ I, x = a, step 3
6	$P(a)$	Assume
7	$\exists y\ [\ Q(y)\ ]$	MP, step 5, 6
8	$Q(b)$	∃ I, y = b, step 7
9	$P(a) \to Q(b)$	→ Add, step 6–8
10	$\exists y\ [\ P(a) \to Q(y)\ ]$	→ Add, step 6–8
11	$\forall x\ \exists y\ [\ P(a) \to Q(y)\ ]$	∀ Add, step 3–10

This takes 7 additional steps. So, yes, if you really understand the logic, you might wonder why it is so painful to write out all the detailed steps, especially when it appears "obvious". We could add rules to make such derivations quicker, though certainly, this would add a lot of clutter to the rules. There are already quite a few proof rules. The reason you follow the rules is to prevent incorrect conclusions. It may be an overly cautious approach, but it is a reasonable approach too.

> **Advice** When learning to do proofs, try not to make up rules where they are not applicable. But be aware of whether the statement you are proving might be true, by knowing what the statement "means"

What does that mean? A person who might shortcut the rules does so because they believe they see something that is true. That's a good insight to have. It's not bad to see that you can prove something without having to prove it. If you stick too rigidly to the rules, you don't anticipate where statements will lead. However, if you don't follow the rules, you begin to make things up. So, as a beginner in proofs, you should stick to the rules.

13 Developing Intuition

When trying to see if a statement can be proven or is false, there are several things you can do. First, it's sometimes worth making real predicates out of the letters. $P(x)$ can seem rather abstract. However, if you think of $P(x)$ as "x is even", then it has more meaning. You should be careful though. Sometimes you might decide something is true because you've made $P(x)$ a specific predicate, such as $even(x)$, and made conclusions on something too specific. That is, if you changed to another predicate, you might realize that the logic doesn't hold for other predicates. Nevertheless, using real predicates can help you to understand what's going on.

The other idea is to use quantifier expansion. For example, suppose you were asked to prove $\forall x\ [\ P(x)\ ] \to \exists x\ [\ P(x)\ ]$, do you think this is true? Try using quantifier expansion. As usual, pick a small domain of, say, 3 elements. Call the elements a, b, c. The expansion says

$$\Big(P(a) \wedge P(b) \wedge P(c)\Big) \to \Big(P(a) \vee P(b) \vee P(c)\Big)$$

If any of $P(a)$, $P(b)$, or $P(c)$ is false, then the hypothesis is false, and the implication is true. If the hypothesis is true, this means all of $P(a)$, $P(b)$, and $P(c)$ must be true. In that case, $P(a) \vee P(b) \vee P(c)$ would also have to be true. So, quantifier expansion gives you some indication that this is a true statement.

Finally, there is the adversary technique. Statements are processed basically left to right. So, whatever the adversary picks for x in $\forall x\ [\ P(x)\ ]$, you can pick that same x for the second part in $\exists x\ [\ P(x)\ ]$.

Developing intuition about logic helps in the long run, so that you aren't so fixated on the proof rules, and not get a feel for when something is correct or not without actually having to go through the entire proof.

14 Proof Strategies

There's roughly three strategies you should use when applying proof rules.

14.1 Apply Existential Instantiation Early

Why should you apply Existential Instantiation early? Because that rule requires you to instantiate to a completely new constant. If you wait too long, then you will have applied, say, Universal Instantiation and introduced a new constants, and you won't be able to use the constant.

Here's a simple example. Suppose you are given $\forall x\ [\ P(x)\ ]$ and $\exists y\ [\ P(y) \to Q(y)\ ]$, and you want to prove $\exists z\ [\ Q(z)\ ]$. Here's what happens if you don't use existential instantiation soon enough.

1	$\forall x\ [\ P(x)\ ]$	Assume
2	$\exists x\ [\ P(y) \to Q(y)\ ]$	Assume
3	$P(a)$	$\forall$ I, step 1, x = a
4	$P(b) \to Q(b)$	$\exists$ I, step 2, y = b

At step 4, you'd really like to instantiate y to a, but you can't because a already appeared in step 3. Instead, you should have existentially instantiated y in step 3, *before* universally instantiating. You can always universally instantiate to any constant, so you might as well wait until you have existentially instantiated first. This is how it should appear:

1	$\forall x\ [\ P(x)\ ]$	Assume
2	$\exists x\ [\ P(y) \rightarrow Q(y)\ ]$	Assume
3	$P(a) \rightarrow Q(a)$	$\exists$ I, step 2, y = a
4	$P(a)$	$\forall$ I, step 1, x = a
5	$Q(a)$	MP, steps 3, 4

By existentially instantiating in step 3, you can then reuse the same constant in the universal instantiation of step 4, and get to the desired conclusion in step 5.

14.2 When Proving Universal, Use Universal Addition

When you have to prove something of the form $\forall x\ [\ P(x)\ ]$, you should be using Universal Addition. Frankly, I'm surprised when I see students trying something else. There are no other real proof rules for proving universal instantiation. Well, there is, and we'll see it in a moment, but this really ought to be your first thought.

I've noticed, in teaching this course, how often I can say "do it this way" and without fail, students will find some other way to do it, almost showing that students honestly prefer to do things their own way, and never listen to advice. Another wonderful observation from me. If you ever listened to all the (positive) advice you were told, you'd probably be doing fantastic in your life. Most people however prefer not to listen to advice (partly because they are poor listeners) rather than follow it.

Why am I venting like this? Because time and again, I offer the same advice, over and over. Time and again, students fail to remember the advice. Time and again, students fail. Simple as that.

When proving a universal of the form, $\forall x\ [\ P(x)\ ]$, what do you do? See, you forgot already. Unbelievable.

14.3 Try Proof By Contradiction

I suppose if I had one rule of thumb to do proofs. Prove everything by contradiction. It's almost always easier to do it that way. Even though I usually say "Prove $\alpha \rightarrow \beta$ by assuming α and proving β", it's merely to establish the standard technique for proving implications. Nevertheless, you can also prove implications by a proof by contradiction. You assume $\sim (\alpha \rightarrow \beta)$ and you get $\alpha \wedge \sim \beta$ (from Negation of $\rightarrow$).

14.4 Use as Few Constants As Possible

You'll find proofs are easier the fewer constants you use. Universal Instantiation allows you to introduce as many constants as you want. However, in reality, you want to use only the constants that have either appeared already, or introduce as few as needed. If you can get away with one constant, use one constant. If you must use at least two constants, try only two. Even if there are three or more *variables*, you may discover you only need to use two constants. Different variables can instantiate to the same constant, assuming you use universal instantiation.