Notes on mathematical logic

James Aspnes

December 13, 2010

Mathematical logic is the discipline that mathematicians invented in the late nineteenth and early twentieth centuries so they could stop talking nonsense. It's the most powerful tool we have for reasoning about things that we can't really comprehend, which makes it a perfect tool for Computer Science.

1 The basic picture

We want to model something we see in reality with something we can fit in our heads. Ideally we drop most of the features of the real thing that we don't care about and keep the parts that we do care about. But there is a second problem: if our model is very big (and the natural numbers are very *very* big), how do we know what we can say about them?

1.1 Axioms, models, and inference rules

One approach is to true to come up with a list of **axioms** that are true statements about the model and a list of **inference rules** that let us derive new true statements from the axioms. The axioms and inference rules together generate a **theory** that consists of all statements that can be constructed from the axioms by applying the inference rules. The rules of the game are that we can't claim that some statement is true unless it's a theorem: something we can derive as part of the theory.

Simple example: All fish are green (axiom). George Washington is a fish (axiom). From "all X are Y" and "Z is X", we can derive "Z is Y" (inference rule). Thus George Washington is green (theorem). Since we can't do anything else with our two axioms and one inference rule, these three statements together form our entire theory about George Washington, fish, greenness, etc.

Theories are attempts to describe **models**. A model is typically a collection of objects and relations between them. For a given theory, there may be many models that are consistent with it: for example, a model that includes both green fishy George Washington and MC 10,000-foot Abraham Lincoln is consistent with the theory above, because it doesn't say anything about Abraham Lincoln.

1.2 Consistency

A theory is **consistent** if it can't prove both P and not-P for any P. Consistency is incredibly important, since all the logics people actually use can prove anything starting from P and not-P.

1.3 What can go wrong

If we throw in too many axioms, you can get an inconsistency: "All fish are green; all sharks are not green; all sharks are fish; George Washington is a shark" gets us into trouble pretty fast.

If we don't throw in enough axioms, we underconstrain the model. For example, the Peano axioms for the natural numbers (see example below) say (among other things) that there is a number 0 and that any number x has a successor $S(x)$ (think of $S(x)$ as $x + 1$). If we stop there, we might have a model that contains only 0, with $S(0) = 0$. If we add in $0 \neq S(x)$ for any x, then we can get stuck at $S(0) = 1 = S(1)$. If we add yet another axiom that says $S(x) = S(y)$ if and only if $x = y$, then we get all the ordinary natural numbers $0, S(0) = 1, S(1) = 2$, etc., but we could also get some extras: say $0', S(0') = 1', S(1') = 0'$. Characterizing the "correct" natural numbers historically took a lot of work to get right, even though we all know what we mean when we talk about them. The situation is of course worse when we are dealing with objects that we don't really understand; here the most we can hope for is to try out some axioms and see if anything strange happens.

Better yet is to use some canned axioms somebody else has already debugged for us. In this respect the core of mathematics acts like a system library—it's a collection of useful structures and objects that are known to work, and (if we are lucky) may even do exactly what we expect.

1.4 The language of logic

The basis of mathematical logic is **propositional logic**, which was essentially invented by Aristotle. Here the model is a collection of statements that are either true or false. There is no ability to refer to actual things; though we might include the statement "George Washington is a fish", from the point of view of propositional logic that is an indivisible atomic chunk of truth or falsehood that says nothing in particular about George Washington or fish. If we treat it as an axiom we can prove the truth of more complicated statements like "George Washington is a fish or 2+2=5" (true since the first part is true), but we can't really deduce much else. Still, this is a starting point.

If we want to talk about things and their properties, we must upgrade to **predicate logic**. Predicate logic adds both **constants** (stand-ins for objects in the model like "George Washington") and **predicates** (stand-ins for properties like "is a fish"). It also lets use **quantify** over variables and make universal statements like "For all x, if x is a fish then x is green." As a bonus, we usually get functions ("$f(x)$ = the number of books George Washington owns about x")

and equality ("George Washington = 12" implies "George Washington + 5 = 17"). This is enough machinery to define and do pretty much all of modern mathematics.

We will discuss both of these logics in more detail below.

1.5 Standard axiom systems and models

Rather than define our own axiom systems and models from scratch, it helps to use ones that have already proven to be (a) consistent and (b) useful. Almost all mathematics fits in one of the following models:

- The natural numbers $\mathbb{N}$. These are defined using the Peano axioms, and if all you want to do is count, add, and multiply, you don't need much else. (If you want to subtract, things get messy.)
- The integers $\mathbb{Z}$. Like the naturals, only now we can subtract. Division is still a problem.
- The rational numbers $\mathbb{Q}$. Now we can divide. But what about $\sqrt{2}$?
- The real numbers $\mathbb{R}$. Now we have $\sqrt{2}$. But what about $\sqrt{(-1)}$?
- The complex numbers $\mathbb{C}$. Now we are pretty much done. But what if we want to talk about more than one complex number at a time?
- The universe of sets. These are defined using the axioms of set theory, and produce a rich collection of sets that include, among other things, structures equivalent to the natural numbers, the real numbers, collections of same, sets so big that we can't even begin to imagine what they look like, and even bigger sets so big that we can't use the axioms to prove whether they exist or not. Fortunately, in Computer Science we can mostly stop with finite sets, which makes life less confusing.
- Various alternatives to set theory, like lambda calculus, category theory or second-order arithmetic. We won't talk about these much, since they generally don't let you do anything you can't do already with sets.

In practice, the usual way to do things is to start with sets and then define everything else in terms of sets: e.g., 0 is the empty set, 1 is a particular set with 1 element, 2 a set with 2 elements, etc., and from here we work our way up to the fancier numbers. The idea is that if we trust our axioms for sets to be consistent, then the things we construct on top of them should also be consistent (although if we are not careful in our definitions they may not be exactly the things we think they are).

2 Propositional logic

Propositional logic is the simplest form of logic. Here the only statements that are considered are *propositions*, which contain no variables. Because propositions contain no variables, they are either always true or always false.

Examples of propositions:

- $2 + 2 = 4$. (Always true).
- $2 + 2 = 5$. (Always false).

Examples of non-propositions:

- $x + 2 = 4$. (May be true, may not be true; it depends on the value of x.)
- $x \cdot 0 = 0$. (Always true, but it's still not a proposition because of the variable.)
- $x \cdot 0 = 1$. (Always false, but not a proposition because of the variable.)

As the last two examples show, it is not enough for a statement to be always true or always false—whether a statement is a proposition or not is a structural property. But if a statement doesn't contain any variables (or other undefined terms), it is a proposition, and as a side-effect of being a proposition it's always true or always false.

2.1 Operations on propositions

Propositions by themselves are pretty boring. So boring, in fact, that logicians quickly stop talking about actual statements and instead haul out placeholder names for propositions like p, q, or r. But we can build slightly more interesting propositions by combining propositions together using various logical connectives, like:

Negation The *negation* of p is written as $\neg p$, or sometimes $-p$ or $\overline{p}$. It has the property that it is false when p is true, and true when p is false.

Or The *or* of two propositions p and q is written as $p \vee q$, and is true as long as at least one, or possibly both, of p and q is true.[1] This is not always the same as what "or" means in English; in English, "or" often is used for *exclusive or* which is not true if both p and q are true. For example, if someone says "You will give me all your money or I will stab you with this table knife", you would be justifiably upset if you turn over all your money and still get stabbed. But a logician would not be at all surprised, because the standard "or" in propositional logic is an *inclusive or* that allows for both outcomes.

[1] The symbol is a stylized V, intended to represent the Latin word *vel*, meaning "or." (Thanks to Noel McDermott for pointing this out.)

Exclusive or If you want to exclude the possibility that both p and q are true, you can use *exclusive or* instead. This is written as $p \oplus q$, and is true precisely when exactly one of p or q is true. Exclusive or is not used in classical logic much, but is important for many computing applications, since it corresponds to addition modulo 2 (see ModularArithmetic) and has nice reversibility properties (e.g. $p \oplus (p \oplus q)$ always has the same truth-value as q).

And The *and* of p and q is written as $p \wedge q$, and is true only when both p and q are true.[2] This is pretty much the same as in English, where "I like to eat ice cream and I own a private Caribbean island" is not a true statement when made by most people even though most people like to eat ice cream. The only complication in translating English expressions into logical *and*s is that logicians can't tell the difference between "and" and "but": the statement "$2 + 2 = 4$ but $3 + 3 = 6$" becomes simply "$(2 + 2 = 4) \wedge (3 + 3 = 6)$."

Implication This is the most important connective for proofs. An implication represents an "if... then" claim. If p implies q, then we write $p \to q$ or $p \Rightarrow q$, depending on our typographic convention and the availability of arrow symbols in our favorite font. In English, $p \Rightarrow q$ is usually rendered as "If p, then q," as in "If you step on your own head, it will hurt." The meaning of $p \Rightarrow q$ is that q is true whenever p is true, and the proposition $p \Rightarrow q$ is true provided (a) p is false (in which case all bets are off), or (b) q is true. In fact, the only way for $p \Rightarrow q$ to be *false* is for p to be true but q to be false; because of this, $p \Rightarrow q$ can be rewritten as $\neg p \vee q$. So, for example, the statements "If $2+2 = 5$, then I'm the Pope", "If I'm the Pope, then $2 + 2 = 4$", and "If $2 + 2 = 4$, then $3 + 3 = 6$", are all true, provided the if/then is interpreted as implication. Normal English usage does not always match this pattern; instead, if/then in normal speech is often interpreted as the much stronger *biconditional* (see below).

Biconditional Suppose that $p \Rightarrow q$ and $q \Rightarrow p$, so that either both p and q are true or both p and q are false. In this case, we write $p \Leftrightarrow q$, $p \Leftrightarrow q$, or $p \Leftrightarrow q$, and say that p holds **if and only if** q holds. The truth of $p \Leftrightarrow q$ is still just a function of the truth or falsehood of p and q; though there doesn't seem any connection between the two sides of the statement, "$2 + 2 = 5$ if and only if I am the Pope" is still true (provided it is not uttered by the Pope). The only way for $p \Leftrightarrow q$ to be false is for one side to be true and one side to be false.

The result of applying any of these operations is called a **compound proposition**.

Here's what all of this looks like when typeset nicely. Note that in some cases there is more than one way to write a compound expression. Which you

[2] This symbol is a stylized A, short for the latin word *atque*, meaning "and."

choose is a matter of personal preference, but you should try to be consistent.

NOT p	$\neg p, \overline{p}$
p OR q	$p \vee q$
p XOR q	$p \oplus q$
p AND q	$p \wedge q$
p implies q	$p \rightarrow q$, $p \Rightarrow q$
p iff q	$p \leftrightarrow q$, $p \Leftrightarrow q$

2.1.1 Precedence

When reading a logical expression, the order of precedence in the absence of parentheses is negation, and, or, implication, biconditional. So $(\neg p \vee q \wedge r \Rightarrow s \Leftrightarrow t)$ is interpreted as $((((\neg p) \vee (q \wedge r)) \Rightarrow s) \Leftrightarrow t)$. *Or* and *and* are associative, so $(p \vee q \vee r)$ is the same as $((p \vee q) \vee r)$ and as $(p \vee (q \vee r))$, and similarly $(p \wedge q \wedge r)$ is the same as $((p \wedge q) \wedge r)$ and as $(p \wedge (q \wedge r))$.

There does not seem to be a standard convention for the precedence of XOR, since logicians don't use it much. There are plausible arguments for putting XOR in between AND and OR, but it's probably safest just to use parentheses.

Implication is not associative, although the convention is that it binds "to the right," so that $a \Rightarrow b \Rightarrow c$ is read as $a \Rightarrow (b \Rightarrow c)$; few people ever remember this, so it is usually safest to put in the parentheses. I personally have no idea what $p \Leftrightarrow q \Leftrightarrow r$ means, so any expression like this should be written with parentheses as either $(p \Leftrightarrow q) \Leftrightarrow r$ or $p \Leftrightarrow (q \Leftrightarrow r)$.

2.2 Truth tables

To define logical operations formally, we give a **truth table**. This gives, for any combination of truth values (true or false, which as computer scientists we often write as 1 or 0) of the inputs, the truth value of the output. So truth tables are to logic what addition tables or multiplication tables are to arithmetic.

Here is a truth table for negation:

p	$\neg p$
0	1
1	0

And here is a truth table for the rest of the logical operators:

p	q	$p \vee q$	$p \oplus q$	$p \wedge q$	$p \Rightarrow q$	$p \Leftrightarrow q$
0	0	0	0	0	1	1
0	1	1	1	0	1	0
1	0	1	1	0	0	0
1	1	1	0	1	1	1

See also RosenBook §§1.1–1.2 or BiggsBook §§3.1–3.3.

We can think of each row of a truth table as a model for propositonal logic, since the only things we can describe in propositional logic are whether particular propositions are true or not. Constructing a truth table corresponds to

generating all possible models. This can be useful if we want to figure out when a particular proposition is true.

2.3 Tautologies and logical equivalence

A compound proposition that is true no matter what the truth-values of the propositions it contains is called a **tautology**. For example, $p \Rightarrow p$, $p \vee \neg p$, and $\neg(p \wedge \neg p)$ are all tautologies, as can be verified by constructing truth tables. If a compound proposition is always false, it's a **contradiction**. The negation of a tautology is a contradiction and vice versa.

The most useful class of tautologies are **logical equivalences**. This is a tautology of the form $X \Leftrightarrow Y$, where X and Y are compound propositions. In this case, X and Y are said to be **logically equivalent** and we can substitute one for the other in more complex propositions. We write $X \equiv Y$ if X and Y are logically equivalent.

The nice thing about logical equivalence is that is does the same thing for Boolean formulas that equality does for algebraic formulas: if we know (for example), that $p \vee \neg p$ is equivalent to 1, and $q \vee 1$ is equivalent to q, we can grind $q \vee p \vee \neg p \equiv q \vee 1 \equiv 1$ without having to do anything particularly clever. (We will need cleverness later when we prove things where the consequent isn't logically equivalent to the premise.)

To prove a logical equivalence, one either constructs a truth table to show that $X \Leftrightarrow Y$ is a tautology, or transforms X to Y using previously-known logical equivalences. Some examples:

- $p \wedge \neg p \equiv 0$: Construct a truth table

p	$\neg p$	$p \wedge \neg p$	0
0	1	0	0
1	0	0	0

  and observe that the last two columns are always equal.

- $p \vee p \equiv p$: Use the truth table

p	$p \vee p$
0	0
1	1

- $p \Rightarrow q \equiv \neg p \vee q$: Again construct a truth table

p	q	$p \Rightarrow q$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

- $\neg(p \vee q) \equiv \neg p \wedge \neg q$: (one of De Morgan's laws; the other is $\neg(p \wedge q) \equiv \neg p \vee \neg q$).

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
0	0	0	**1**	1	1	**1**
0	1	1	**0**	1	0	**0**
1	0	1	**0**	0	1	**0**
1	1	1	**0**	0	0	**0**

- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ (one of the distributive laws; the other is $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$).

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$
0	0	0	0	**0**	0	0	**0**
0	0	1	0	**0**	1	0	**0**
0	1	0	0	**0**	0	1	**0**
0	1	1	1	**1**	1	1	**1**
1	0	0	0	**1**	1	1	**1**
1	0	1	0	**1**	1	1	**1**
1	1	0	0	**1**	1	1	**1**
1	1	1	1	**1**	1	1	**1**

- $(p \Rightarrow r) \vee (q \Rightarrow r) \equiv (p \wedge q) \Rightarrow r$. Now things are getting messy, so building a full truth table may take awhile. But we have take a shortcut by using logical equivalences that we've already proved (plus associativity of $\vee$):

$$\begin{aligned}
(p \Rightarrow r) \vee (q \Rightarrow r) &\equiv (\neg p \vee r) \vee (\neg q \vee r) && \text{[Using } p \Rightarrow q \equiv \neg p \vee q \text{ twice]} \\
&\equiv \neg p \vee \neg q \vee r \vee r && \text{[Associativity and commutativity of } \vee] \\
&\equiv \neg p \vee \neg q \vee r && [p \equiv p \vee p] \\
&\equiv \neg(p \wedge q) \vee r && \text{[De Morgan's law]} \\
&\equiv (p \wedge q) \Rightarrow r. && [p \Rightarrow q \equiv \neg p \vee q]
\end{aligned}$$

This last equivalence is a little surprising. It shows, for example, that if somebody says "It is either the case that if you study you will graduate from Yale with distinction, or that if you join the right secret society you will graduate from Yale with distinction", then this statement (assuming we treat the or as $\vee$) is logically equivalent to "If you study and join the right secret society, then you will graduate from Yale with distinction." It is easy to get tangled up in trying to parse the first of these two propositions; translating to logical notation and simplifying using logical equivalence is a good way to simplify it.

Over the years, logicians have given names to many logical equivalences. Only a few of these are actually useful to remember. These include **de Morgan's laws**: $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ and $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ (see WikiPedia's page on the subject for more versions of these), commutativity of AND and OR, the equivalence of $p \Rightarrow q$ and $\neg p \vee q$, and the contraposition rule $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$ (see below for more about contraposition). Anything else you need you can probably derive from these or by writing out a truth table.

2.4 Inverses, converses, and contrapositives

The **inverse** of $p \Rightarrow q$ is $\neg p \Rightarrow \neg q$. So the inverse of "If you take CPSC 202, you will surely die" is "If you do not take CPSC 202, you will not surely die." There is often no connection between the truth of an implication and the truth of its inverse: "If I am Barack Obama then I am a Democrat" does not have the same truth-value as "If I am not Barack Obama then I am not a Democrat", at least according to current polling numbers.

The **converse** of $p \Rightarrow q$ is $q \Rightarrow p$. E.g. the converse of "If I am Barack Obama then I am a Democrat" is "If I am a Democrat then I am Barack Obama." The converse of a statement is always logically equivalent to the inverse. Often in proving a biconditional (e.g., "I am Barack Obama if and only if I am a Democrat"), one proceeds by proving first the implication in one direction and then either the inverse or the converse.

The **contrapositive** of $p \Rightarrow q$ is $\neg q \Rightarrow \neg p$; it is logically equivalent to the original implication ("If I am not a Democrat then I am not Barack Obama"). A **proof by contraposition** proves p implies q by assuming $\neg q$ and then proving $\neg p$; it is similar but not identical to an **indirect proof**, which assumes $\neg p$ and derives a contradiction.

2.5 Normal forms

A compound proposition is in **conjuctive normal form** (**CNF** for short) if it is obtained by ANDing together ORs of one or more variables or their negations (an OR of one variable is just the variable itself). So for example P, $(P \vee Q) \wedge R$, $(P \vee Q) \wedge (Q \vee R) \wedge (\neg P)$, and $(P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \vee Q \vee S \vee T \vee \neg U)$ are in CNF, but $(P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \wedge Q)$, $(P \vee Q) \wedge (P \Rightarrow R) \wedge (\neg P \vee Q)$, and $(P \vee (Q \wedge R)) \wedge (P \vee \neg R) \wedge (\neg P \vee Q)$ are not. Using the equivalence $P \Rightarrow Q \equiv \neg P \vee Q$, De Morgan's laws, and the distributive law, it is possible to rewrite any compound proposition in CNF.

CNF formulas are particularly useful because they support **resolution** (see *Inference rules* below). Using the tautology $(P \vee Q) \wedge (\neg P \vee R) \Rightarrow Q \vee R$, we can construct proofs from CNF formulas by looking for occurrences of some simple proposition and its negation and **resolving** them, which generates a new clause we can add to the list. For example, we can compute

$$\begin{aligned}(\mathbf{P} \vee Q) \wedge (P \vee \neg R) \wedge (\neg \mathbf{P} \vee Q) \wedge (\neg Q \vee R) &\vdash (P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \vee Q) \wedge (\neg \mathbf{Q} \vee R) \wedge \mathbf{Q} \\ &\vdash (P \vee Q) \wedge (P \vee \neg \mathbf{R}) \wedge (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge Q \wedge \mathbf{R} \\ &\vdash (P \vee Q) \wedge (P \vee \neg R) \wedge (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge Q \wedge R \wedge P \\ &\vdash P.\end{aligned}$$

This style of proof is called a **resolution proof**. Because of its simplicity it is particularly well-suited for mechanical theorem provers. Such proofs can also encode traditional proofs based on *modus ponens*: the inference $P \wedge (P \Rightarrow Q) \vdash Q$ can be rewritten as resolution by expanding $\Rightarrow$ to get $P \wedge (\neg P \vee Q) \vdash Q$.

Similarly, a compound proposition is in **disjunctive normal form** (**DNF**) if it consists of an OR of ANDs, e.g. $(P \wedge Q) \vee (P \wedge \neg R) \vee (\neg P \wedge Q)$. Just as any compound proposition can be transformed into CNF, it can similarly be transformed into DNF.

Note that conjunctive normal forms are not unique; for example, $P \vee Q$ and $(P \wedge \neg Q) \vee (P \wedge Q) \vee (\neg P \wedge Q)$ are both in conjunctive normal form and are logically equivalent to each other. So while CNF can be handy as a way of reducing the hairiness of a formula (by eliminating nested parentheses or negation of non-variables, for example), it doesn't necessarily let us see immediately if two formulas are really the same.

3 Predicate logic

Using only propositional logic, we can express a simple version of a famous argument:

- Socrates is a man.
- If Socrates is a man, then Socrates is mortal.
- Therefore, Socrates is mortal.

This is an application of the inference rule called *modus ponens*, which says that from p and $p \Rightarrow q$ you can deduce q. The first two statements are axioms (meaning we are given them as true without proof), and the last is the conclusion of the argument.

What if we encounter Socrates's infinitely more logical cousin Spocrates? We'd like to argue

- Spocrates is a man.
- If Spocrates is a man, then Spocrates is mortal.
- Therefore, Spocrates is mortal.

Unfortunately, the second step depends on knowing that humanity implies mortality for everybody, not just Socrates. If we are unlucky in our choice of axioms, we may not know this. What we would like is a general way to say that humanity implies mortality for everybody, but with just propositional logic, we can't write this fact down.

3.1 Variables and predicates

The solution is to extend our language to allow formulas that involve variables. So we might let x, y, z, etc. stand for any element of our **universe of discourse** or **domain**—essentially whatever things we happen to be talking about at the moment. We can now write statements like:

- "x is human."
- "x is the parent of y."
- "$x + 2 = x^2$."

These are not propositions because they have variables in them. Instead, they are **predicates**; statements whose truth-value depends on what concrete object takes the place of the variable. Predicates are often abbreviated by single capital letters followed by a list of **arguments**, the variables that appear in the predicate, e.g.:

- $H(x)$ = "x is human."
- $P(x, y)$ = "x is the parent of y."
- $Q(x)$ = "x + 2 = x^2."

We can also fill in specific values for the variables, e.g. H(Spocrates) = "Spocrates is human." If we fill in specific values for all the variables, we have a proposition again, and can talk about that proposition being true (e.g. $H(2)$ and $H(-1)$ are true) or false ($H(0)$ is false).

3.2 Quantifiers

What we really want though is to be able to say when H or P or Q is true for many different values of their arguments. This means we have to be able to talk about the truth or falsehood of statements that include variables. To do this, we **bind** the variables using **quantifiers**, which state whether the claim we are making applies to all values of the variable (**universal quantification**), or whether it may only apply to some (**existential quantification**).

3.2.1 Universal quantifier

The **universal quantifier** $\forall$ (pronounced "for all") says that a statement must be true for all values of a variable within some *universe* of allowed values (which is often implicit). For example, "all humans are mortal" could be written $\forall x : \text{Human}(x) \Rightarrow \text{Mortal}(x)$ and "if x is positive then $x + 1$ is positive" could be written $\forall x : x > 0 \Rightarrow x + 1 > 0$.

If you want to make the universe explicit, use set membership notation, e.g. $\forall x \in Z : x > 0 \Rightarrow x + 1 > 0$. This is logically equivalent to writing $\forall x : x \in Z \Rightarrow (x > 0 \Rightarrow x+1 > 0)$ or to writing $\forall x : (x \in Z \wedge x > 0) \Rightarrow x+1 > 0$, but the short form makes it more clear that the intent of $x \in Z$ is to restrict the range of x.[3]

The statement $\forall x : P(x)$ is equivalent to a very large AND; for example, $\forall x \in \mathbb{N} : P(x)$ could be rewritten (if you had an infinite amount of paper) as $P(0) \wedge P(1) \wedge P(2) \wedge P(3) \wedge \ldots$. Normal first-order logic doesn't allow infinite

[3]Programmers will recognize this as a form of *syntactic sugar*.

expressions like this, but it may help in visualizing what $\forall x : P(x)$ actually means. Another way of thinking about it is to imagine that x is supplied by some adversary and you are responsible for showing that $P(x)$ is true; in this sense, the universal quantifier chooses the *worst case* value of x.

3.2.2 Existential quantifier

The **existential quantifier** $\exists$ (pronounced "there exists") says that a statement must be true for at least one value of the variable. So "some human is mortal" becomes $\exists x : \text{Human}(x) \wedge \text{Mortal}(x)$. Note that we use AND rather than implication here; the statement $\exists x : \text{Human}(x) \Rightarrow \text{Mortal}(x)$ makes the much weaker claim that "there is some thing x, such that if x is human, then x is mortal," which is true in any universe that contains an immortal purple penguin—since it isn't human, Human(penguin) $\Rightarrow$ Mortal(penguin) is true.

As with $\forall$, $\exists$ can be limited to an explicit universe with set membership notation, e.g., $\exists x \in Z : x = x^2$. This is equivalent to writing $\exists x : x \in Z \wedge x = x^2$.

The formula $\exists x : P(x)$ is equivalent to a very large OR, so that $\exists x \in \mathbb{N} : P(x)$ could be rewritten as $P(0) \vee P(1) \vee P(2) \vee P(3) \vee \ldots$. Again, you can't generally write an expression like this if there are infinitely many terms, but it gets the idea across.

3.2.3 Negation and quantifiers

- $\neg \forall x : P(x) \equiv \exists x : \neg P(x)$.
- $\neg \exists x : P(x) \equiv \forall x : \neg P(x)$.

These are essentially the quantifier version of De Morgan's laws: the first says that if you want to show that not all humans are mortal, it's equivalent to finding some human that is not mortal. The second says that to show that no human is mortal, you have to show that all humans are not mortal.

3.2.4 Restricting the scope of a quantifier

Sometimes we want to limit the universe over which we quantify to some restricted set, e.g., all positive integers or all baseball teams. We can do this explicitly using implication:

$$\forall x : x > 0 \Rightarrow x - 1 \geq 0$$

or we can abbreviate by including the restriction in the quantifier expression itself:

$$\forall x > 0 : x - 1 \geq 0.$$

Similarly

$$\exists x : x > 0 \wedge x^2 = 81$$

can be written as

$$\exists x > 0 : x^2 = 81.$$

Note that constraints on $\exists$ get expressed using AND rather than implication.

The use of set membership notation to restrict a quantifier is a special case of this. Suppose we want to say that 79 is not a perfect square, by which we mean that there is no integer whose square is 79. If we are otherwise talking about real numbers (two of which happen to be square roots of 79), we can exclude the numbers we don't want by writing

$$\neg\exists x \in \mathbb{Z} x^2 = 79$$

which is interpreted as

$$\neg\exists x(x \in \mathbb{Z} \wedge x^2 = 79)$$

or, equivalently

$$\forall x x \in \mathbb{Z} \Rightarrow x^2 \neq 79.$$

Here $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is the standard set of integers. See also SetTheory for more about $\in$.

3.2.5 Nested quantifiers

It is possible to nest quantifiers, meaning that the statement bound by a quantifier itself contains quantifiers. For example, the statement "there is no largest prime number" could be written as

$$\neg\exists x : (\text{Prime}(x) \wedge \forall y : y > x \Rightarrow \neg\text{Prime}(y))$$

i.e., "there does not exist an x that is prime and any y greater than x is not prime." Or in a shorter (though not strictly equivalent) form:

$$\forall x \exists y : y > x \wedge \text{Prime}(y)$$

which we can read as "for any x there is a bigger y that is prime."

To read a statement like this, treat it as a game between the $\forall$ player and the $\exists$ player. Because the $\forall$ comes first in this statement, the for-all player gets to pick any x it likes. The $\exists$ player then picks a y to make the rest of the statement true. The statement as a whole is true if the $\exists$ player always wins the game. So if you are trying to make a statement true, you should think of the universal quantifier as the enemy (the **adversary** in algorithm analysis) who says "nya-nya: try to make *this* work, bucko!", while the existential quantifier is the friend who supplies the one working response.

Naturally, such games can go on for more than two steps, or allow the same player more than one move in a row. For example

$$\forall x \forall y \exists z x^2 + y^2 = z^2$$

is a kind of two-person challenge version of the Pythagorean theorem where the universal player gets to pick x and y and the existential player has to respond with a winning z. (Whether the statement itself is true or false depends on the range of the quantifiers; it's false, for example, if x, y, and z are all natural

numbers or rationals but true if they are all real or complex. Note that the universal player only needs to find one bad (x, y) pair to make it false.)

One thing to note about nested quantifiers is that we can switch the order of two universal quantifiers or two existential quantifiers, but we can't swap a universal quantifier for an existential quantifier or vice versa. So for example $\forall x \forall y (x = y \Rightarrow x + 1 = y + 1)$ is logically equivalent to $\forall y \forall x (x = y \Rightarrow y + 1 = x + 1)$, but $\forall x \exists y y < x$ is not logically equivalent to $\exists y \forall x y < x$. This is obvious if you think about it in terms of playing games: if I get to choose two things in a row, it doesn't really matter which order I choose them in, but if I choose something and then you respond it might make a big difference if we make you go first instead.

One measure of the complexity of a mathematical statement is how many layers of quantifiers it has, where a layer is a sequence of all-universal or all-existential quantifiers. Here's a standard mathematical definition that involves three layers of quantifiers, which is about the limit for most humans:

$$\begin{aligned} & \lim_{x \to \infty} f(x) = y \\ \equiv \quad & \forall \epsilon > 0 : \exists N : \forall x > N : |f(x) - y| < \epsilon. \end{aligned}$$

Now that we know how to read nested quantifiers, it's easy to see what the right-hand side means:

1. The adversary picks ϵ, which must be greater than 0.
2. We pick N.
3. The adversary picks x, which must be greater than N.
4. We win if $f(x)$ is within ϵ of y.

So, for example, a proof of

$$\lim_{x \to \infty} 1/x = 0$$

would follow exactly this game plan:

1. Choose some $\epsilon > 0$.
2. Let $N > 1/\epsilon$. (Note that we can make our choice depend on previous choices.)
3. Choose any $x > N$.
4. Then $x > N > 1/\epsilon > 0$, so $1/x < 1/N < \epsilon \Rightarrow |1/x - 0| < \epsilon$. QED!

3.2.6 Examples

Here are some more examples of translating English into statements in predicate logic:

English	Logic
All crows are black.	$\forall x \text{Crow}(x) \Rightarrow \text{Black}(x)$
Some cows are brown.	$\exists x Cow(x) \wedge Brown(x)$
No cows are blue.	$\neg\exists x Cow(x) \wedge Blue(x)$
All that glitters is not gold.	$\neg\forall x \text{Glitters}(x) \Rightarrow \text{Gold}($
No shirt, no service.	$\forall x \neg Shirt(x) \Rightarrow \neg Serve$
Every event has a cause.	$\forall x \exists y \text{Causes}(y, x)$
Every even number greater than 2 can be expressed as the sum of two primes.	$\forall x(Even(x) \wedge x > 2) \Rightarrow$

3.3 Functions

A **function symbol** looks like a predicate but instead of computing a truth value it returns a new object. So for example the successor function S in the Peano axioms for the natural numbers returns $x + 1$ when applied as $S(x)$. Sometimes when there is only a single argument we omit the parentheses, e.g., $Sx = S(x), SSSx = S(S(S(x)))$.

3.4 Equality

Often we include a special **equality** predicate $=$, written $x = y$. The interpretation of $x = y$ is that x and y are the same element of the domain. It satisfies the **reflexivity axiom** $\forall x : x = x$ and the **substitution axiom schema**:

$$\forall x \forall y(x = y \Rightarrow (Px \Leftrightarrow Py))$$

where P is any predicate. This immediately gives a **substitution rule** that says $x = y, P(x) \vdash P(y)$. It's likely that almost every proof you ever wrote down in high school algebra consisted only of many applications of the substitution rule.

Example: We'll prove $\forall x \forall y(x = y \Rightarrow y = x)$ from the above axioms (this property is known as **symmetry**). Apply substitution to the predicate $Px \equiv y = x$ to get $\forall x \forall y(x = y \Rightarrow (y = x \Leftrightarrow x = x))$. Use reflexivity to rewrite this as $\forall x \forall y(x = y \Rightarrow (y = x \Leftrightarrow T))$ or $\forall x \forall y(x = y \Rightarrow y = x)$ as claimed.

Exercise: Prove $\forall x \forall y \forall z(x = y \wedge y = z \Rightarrow x = z)$. (This property is known as **transitivity**.)

3.4.1 Uniqueness

An occasionally useful abbreviation is $\exists! x P(x)$, which stands for "there exists a *unique* x such that $P(x)$." This is short for

$$(\exists x P(x)) \wedge (\forall x \forall y P(x) \wedge P(y) \Rightarrow x = y).$$

An example is $\exists! x x + 1 = 12$. To prove this we'd have to show not only that there is some x for which $x + 1 = 12$ (11 comes to mind), but that if we have any two values x and y such that $x + 1 = 12$ and $y + 1 = 12$, then $x = y$ (this is not hard to do). So the exclamation point encodes quite a bit of extra work, which is why we usually hope that $\exists x x + 1 = 12$ is good enough.

3.5 Models

In propositional logic, we can build truth tables that describe all possible settings of the truth-values of the literals. In predicate logic, the analogous concept to an assignment of truth-values is a **structure**. A structure consists of a set of objects or elements (see **set theory**), together with a description of which elements fill in for the constant symbols, which predicates hold for which elements, and what the value of each function symbol is when applied to each possible list of arguments (note that this depends on knowing what constant, predicate, and function symbols are available—this information is called the **signature** of the structure). A structure is a **model** of a particular **theory** (set of statements), if each statement in the theory is true in the model.

In general we can't hope to find all possible models of a given theory. But models are useful for two purposes: if we can find some model of a particular theory, then the existence of this model demonstrates that the theory is consistent; and if we can find a model of the theory in which some additional statement S doesn't hold, then we can demonstrate that there is no way to prove S from the theory (i.e. it is not the case that $T \vdash S$, where T is the list of axioms that define the theory).

3.5.1 Examples

- Consider the axiom $\neg\exists x$. This axiom has exactly one model (it's empty).
- Now consider the axiom $\exists! x$. This axiom also has exactly one model (with one element).
- We can enforce exactly k elements with one rather long axiom, e.g. for $k = 3$ do $\exists x_1 \exists x_2 \exists x_3 \forall y y = x_1 \vee y = x_2 \vee y = x_3$. In the absence of any special symbols, a structure of 3 undifferentiated elements is the unique model of this axiom.
- Suppose we add a predicate P and consider the axiom $\exists x P x$. Now we have many models: take any nonempty model you like, and let P be true of at least one of its elements. If we take a model with two elements a and b, with Pa and $\neg Pb$, we get that $\exists x P x$ is not enough to prove $\forall x P x$, since the later statement isn't true in this model.
- Now let's bring in a function symbol S and constant symbol 0. Consider a stripped-down version of the Peano axioms that consists of just the axiom $\forall x \forall y S x = S y \Rightarrow x = y$. Both the natural numbers $\mathbb{N}$ and the integers

$\mathbb{Z}$ are a model for this axiom, as is the set $\mathbb{Z}_m$ of integers mod m for any m (see ModularArithmetic). In each case each element has a unique predecessor, which is what the axiom demands. If we throw in the first Peano axiom $\forall x Sx \neq 0$, we eliminate $\mathbb{Z}$ and $\mathbb{Z}_m$ because in each of these models 0 is a successor of some element. But we don't eliminate a model that consists of two copies of $\mathbb{N}$ sitting next to each other (only one of which contains the "real" 0), or even a model that consists of one copy of $\mathbb{N}$ (to make 0 happy) plus any number of copies of $\mathbb{Z}$ and $\mathbb{Z}_m$.

- A practical example: The family tree of the kings of France is a model of the theory containing the two axioms $\forall x \forall y \forall z \text{Parent}(x, y) \wedge \text{Parent}(y, z) \Rightarrow \text{GrandParent}(x, z)$ and $\forall x \forall y \text{Parent}(x, y) \Rightarrow \neg \text{Parent}(y, x)$. But this set of axioms could use some work, since it still allows for the possibility that $\text{Parent}(x, y)$ and $\text{GrandParent}(y, x)$ are both true.

4 Proofs

A proof is a way to derive statements from other statements. It starts with *axioms* (statements that are assumed in the current context always to be true), *theorems* or *lemmas* (statements that were proved already), and *premises* P, and uses **inference rules** to derive Q. The axioms, theorems, and premises are in a sense the starting position of a game whose rules are given by the inference rules. The goal of the game is to apply the inference rules until Q pops out. We refer to anything that isn't proved in the proof itself (i.e., an axiom, theorem, lemma, or premise) as a **hypothesis**; the result Q is the **conclusion**.

When a proof exists of Q from some premises $P_1, P_2, \ldots$, we say that Q is **deducible** or **provable** from $P_1, P_2, \ldots$, which is written as

$$P_1, P_2, \ldots \vdash Q.$$

Note that axioms are usually not included in list on the left-hand side. So we can write just

$$\vdash Q$$

if we can prove Q directly from our axioms without any additional premises.

The **turnstile** symbol $\vdash$ has the specific meaning that we can derive the conclusion Q by applying inference rules to the premises. This is not quite the same thing as saying $P \Rightarrow Q$. If our inference rules are particularly weak, it may be that $P \Rightarrow Q$ is true but we can't prove Q starting with P. Conversely, if our inference rules are too strong (maybe they can prove anything, even things that aren't true) we might have $P \vdash Q$ but $P \Rightarrow Q$ is false. But most of the time we will use inference rules that are just right, meaning that $P \vdash Q$ implies $P \Rightarrow Q$ (**soundness**) and $P \Rightarrow Q$ implies $P \vdash Q$ (**completeness**). The distinction between $\vdash$ and $\Rightarrow$ is then whether we want to talk about the existence of a proof (the first case) or about the logical relation between two statements (the second). As a consequence, you probably won't see $\vdash$ again after you stop reading these notes.

4.1 Inference Rules

The main source of inference rules is tautologies of the form $P \Rightarrow Q$; given such a tautology, there is a corresponding inference rule that allows us to assert Q once we have P (either because it's an axiom/theorem/premise or because we proved it already while doing the proof). The most important inference rule is **modus ponens**, based on the tautology $(p \wedge (p \Rightarrow q)) \Rightarrow q$; this lets us, for example, write the following famous argument:[4]

1. If it doesn't fit, you must acquit. [Axiom]
2. It doesn't fit. [Premise]
3. You must acquit. [Modus ponens applied to 1+2]

There are many named inference rules in classical propositional logic. We'll list some of them below. You don't need to remember the names of anything except modus ponens, and most of the rules are pretty much straightforward applications of modus ponens plus some convenient tautology that can be proved by truth tables or stock logical equivalences. (For example, the "addition" rule below is just the result of applying modus ponens to p and the tautology $p \Rightarrow (p \vee q)$.)

Inference rules are often written by putting the premises above a horizontal line and the conclusion below. In text, the horizontal line is often replaced by the symbol $\vdash$, which means exactly the same thing. Premises are listed on the left-hand side separated by commas, and the conclusion is placed on the right. We can then write

Addition $p \vdash p \vee q$.

Simplification $p \wedge q \vdash p$.

Conjunction $p, q \vdash p \wedge q$.

Modus ponens $p, p \Rightarrow q \vdash q$.

Modus tollens $\neg q, p \Rightarrow q \vdash \neg p$.

Hypothetical syllogism $p \Rightarrow q, q \Rightarrow r \vdash p \Rightarrow r$.

Disjunctive syllogism $p \vee q, \neg p \vdash q$.

Resolution $p \vee q, \neg p \vee r \vdash q \vee r$.

Of these rules, Addition, Simplification, and Conjunction are mostly used to pack and unpack pieces of arguments. Modus ponens (and its reversed cousin modus tollens) let us apply implications. You only need to remember modus ponens if you can remember the contraposition rule $(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$. Hypothetical syllogism just says that implication is transitive; it lets you paste

[4] OK, maybe not as famous as it once was.

together implications if the conclusion of one matches the premise of the other. Disjunctive syllogism is again a disguised version of modus ponens (via the logical equivalence $(p \vee q) \equiv (\neg p \Rightarrow q)$); you don't need to remember it if you can remember this equivalence. Resolution is almost never used by humans but is very popular with computer theorem provers.

An argument is **valid** if the conclusion is true whenever the hypotheses are true. Any proof constructed using the inference rules is valid. It does not necessarily follow that the conclusion is true; it could be that one or more of the hypotheses is false:

1. If you give a mouse a cookie, he's going to ask for a glass of milk. [Axiom]
2. If he asks for a glass of milk, he will want a straw. [Axiom]
3. You gave a mouse a cookie. [Premise]
4. He asks for a glass of milk. [Modus ponens applied to 1 and 3.]
5. He will want a straw. [Modus ponens applied to 2 and 4.]

Will the mouse want a straw? No: Mice can't ask for glasses of milk, so axiom 1 is false.

4.2 More on proofs vs implication

Recall that $P \vdash Q$ means there is a proof of Q by applying inference rules to the axioms and P, while $P \Rightarrow Q$ says that Q holds whenever P does. These are not the same thing: provability ($\vdash$) is outside the theory (it's a statement about whether a proof exists or not) while implication ($\Rightarrow$) is inside (it's a logical connective for making compound propositions). But most of the time they mean almost the same thing.

For example, suppose that we can prove $P \Rightarrow Q$ directly from whatever axioms we are currently assuming, i.e., that

$$\vdash P \Rightarrow Q.$$

Since we can always ignore extra premises, we get

$$P \vdash P \Rightarrow Q$$

and thus

$$P \vdash P, P \Rightarrow Q$$

and finally

$$P \vdash Q$$

by applying modus ponens to the right-hand side.

So we can go from $\vdash P \Rightarrow Q$ to $P \vdash Q$.

This means that provability is in a sense weaker than implication: it holds (assuming modus ponens) whenever implication does. But we usually don't use this fact much, since $P \Rightarrow Q$ is a much more useful statement than $P \vdash Q$. Can we go the other way?

4.2.1 The Deduction Theorem

Yes, using the **Deduction Theorem**.

Often we want to package the result of a proof as a Theorem (a proven statement that is an end in itself) or Lemma (a proven statement that is intended mostly to be used in other proofs). Typically a proof shows that if certain premises $P_1, P_2, \ldots P_n$ hold, then some conclusion Q holds (with various axioms or previously-established theorems assumed to be true from context). To use this result later, it is useful to be able to package it as an implication $P_1 \wedge P_2 \wedge \ldots P_n \Rightarrow Q$. In other words, we want to go from

$$P_1, P_2, \ldots, P_n \vdash Q$$

to

$$\vdash (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \Rightarrow Q.$$

The statement that we can do this, for a given set of axioms and inference rules, is the

Theorem 1 (Deduction Theorem)**.** *If there is a proof of Q from premises $P_1, P_2, \ldots, P_n$, then there is a proof of $P_1 \wedge P_2 \wedge \ldots \wedge P_n \Rightarrow Q$.*

The actual proof of the theorem depends on the particular set of axioms and inference rules we start with, but the basic idea is that there exists a mechanical procedure for extracting a proof of the implication from the proof of Q assuming P_1 etc.

Caveat : In predicate logic, the deduction theorem only applies if none of the premises contain any free variables (which are variables that aren't bound by a containing quantifier). Usually you won't run into this, but see here for some bad cases that arise without this restriction.

4.3 Inference rules for equality

The equality predicate is special, in that it allows for the **substitution rule**

$$x = y, P(x) \vdash P(y).$$

This can also be represented as an axiom schema: $\forall x : \forall y : ((x = y \wedge P(x)) \Rightarrow P(y))$.

4.4 Inference rules for quantified statements

Universal generalization If $P \vdash Q(x)$ and x does not appear in P, then $P \vdash \forall x : Q(x)$. Typical use: We want to show that there is no biggest natural number, i.e. that $\forall x \in \mathbb{N} : \exists y \in \mathbb{N} : y > x$. Let x be any element of $\mathbb{N}$. Let $y = x + 1$. Then $y > x$. (Note: there is also an instance of existential generalization here.)

Universal instantiation $\forall x Q(x) \vdash Q(c)$. Typical use: Given that all humans are mortal, it follows that Spocrates is mortal.

Existential generalization $Q(c) \vdash \exists x Q(x)$. Typical use: We are asked to prove that there exists an even prime number. Look at 2: it's an even prime number. QED.

Existential instantiation $\exists x Q(x) \vdash Q(c)$ for some particular c. (The real rule is more complicated and says $((\forall x : (Q(x) \Rightarrow P)) \wedge \exists y : Q(y)) \Rightarrow P$; but the intent in both cases is that once you have proven that at least one c satisfying Q exists, you can give it a name and treat it as an object you actually have in your hands.) Typical use: Suppose we are told that there exists a prime number greater than 7. Let p be some such prime number greater than 7.

5 Proof techniques

A proof technique is a template for how to go about proving particular classes of statements: this template guides you in the choice of inference rules (or other proof techniques) to write the actual proof. This doesn't substitute entirely for creativity (there is no efficient mechanical procedure for generating even short proofs unless), but it can give you some hints for how to get started.

In the table below, we are trying to prove $A \Rightarrow B$ for particular statements A and B. The techniques are mostly classified by the structure of B. Before applying each technique, it may help to expand any definitions that appear in A or B.[5]

If you want to prove $A \Leftrightarrow B$, the usual approach is to prove $A \Rightarrow B$ and $A \Leftarrow B$ separately; proving $A \Rightarrow B$ and $\neg A \Rightarrow \neg$B also works (because of contraposition).

Strategy	When to use it	What to ass
Direct proof	Try it first	A
Contraposition	$B = \neg Q$	$\neg B$
Contradiction	When $B = \neg Q$, or when you are stuck trying the other techniques.	$A \wedge \neg B$
Construction	$B = \exists x P(x)$	A
Counterexample	$B = \neg \forall x P(x)$	A
Choose	$B = \forall x (P(x) \Rightarrow Q(x))$	A and $P(c)$
Instantiation	$A = \forall x P(x)$	A
Proof by elimination	$B = C \vee D$	$A \wedge \neg C$
Case analysis	$A = C \vee D$	C, D
Induction	$B = \forall x \in \mathbb{N} P(x)$	A

[5]These strategies are largely drawn from SolowBook, particularly the summary table on pages 156–159 (which is the source of the order and organization of the table and the names of most of the techniques). RosenBook describes proof strategies of various sorts in §§1.5–1.7 and BiggsBook describes various proof techniques in Chapters 1, 3, and 4; both descriptions are a bit less systematic, but also include a variety of specific techniques that are worth looking at.

Some others that are mentioned in SolowBook: Direct Uniqueness, Indirect Uniqueness, various max/min arguments. We will also cover induction proofs in more detail later.

6 Example: The natural numbers

Here we give an example of how we can encode simple mathematics using predicate logic, and then prove theorems about the resulting structure. Our goal is to represent the **natural numbers**: 0, 1, 2, etc.[6]

6.1 The Peano axioms

Peano represents natural numbers using a special 0 constant and a function symbol S (for "successor"; think of it as $+1$). Repeatedly applying S to 0 generates increasingly large natural numbers: $S0 = 1, SS0 = 2, SSS0 = 3$, etc. (Note that 1, 2, 3, etc., are not part of the language, although we might use them sometimes as shorthand for long strings of S's.) For convenience, we don't bother writing parentheses for the function applications unless we need to do so to avoid ambiguity: read $SSSS0$ as $S(S(S(S(0))))$.

The usual interpretation of function symbols implies that 0 and its successors exist, but it doesn't guarantee that they aren't all equal to each other. The first Peano axiom prevents this:

$$\forall x Sx \neq 0.$$

In English, 0 is not the successor of any number.

This still allows for any number of nasty little models in which 0 is nobody's successor, but we still stop before getting all of the naturals. For example, let $SS0 = S0$; then we only have two elements in our model (0 and $S0$, because once we get to $S0$, any further applications of S keep us where we are.

To avoid this, we need to prevent S from looping back round to some number we've already produced. It can't get to 0 because of the first axiom, and to prevent it from looping back to a later number, we take advantage of the fact that they already have one successor:

$$\forall x : \forall y : Sx = Sy \Rightarrow x = y.$$

If we take the contrapositive in the middle, we get $x \neq y \Rightarrow Sx \neq Sy$. In other words, we can't have a single number z that is the successor of two different numbers x and y.

Now we get all of $\mathbb{N}$, but we may get some extra elements as well. There is nothing in the first two axioms that prevents us from having something like this:

$$0 \to S0 \to SS0 \to SSS0 \to \ldots B \to SB \to SSB \to SSSB \to \ldots$$

[6] Some people define the natural numbers as starting at 1. Those people are generally either (a) wrong, (b) number theorists, (c) extremely conservative, or (d) citizens of the United Kingdom of Great Britain and Northern Ireland. As computer scientists, we will count from 0 as the gods intended.

where B stands for "bogus."

The hard part of coming up with the Peano axioms was to prevent the model from sneaking in extra bogus values (that still have successors and at most one predecessor each). This is (almost) done using the third Peano axiom, which in modern terms is actually an **axiom schema**: an axiom that holds when substituting any predicate for its placeholder predicate P:

$$(P(0) \wedge (\forall x : P(x) \Rightarrow P(Sx))) \Rightarrow \forall x : P(x).$$

This is known as the **induction schema**, and says that, for any predicate P, if we can prove that P holds for 0, and we can prove that $P(x)$ implies $P(x+1)$, then P holds for all x in $\mathbb{N}$. The intuition is that even though we haven't bothered to write out a proof of, say $P(1337)$, we know that we can generate one by starting with $P(0)$ and modus-pwning our way out to $P(1337)$ using $P(0) \Rightarrow P(1)$, then $P(1) \Rightarrow P(2)$, then $P(2) \Rightarrow P(3)$, etc. Since this works for any number (eventually), there can't be some number that we missed.

In particular, this lets us throw out the bogus numbers in the bad example above. Let $B(x)$ be true if x is bogus (i.e., it's equal to B or one of the other values in its chain of successors). Let $P(x) \equiv \neg B(x)$. Then $P(0)$ holds (0 is not bogus), and if $P(x)$ holds (x is not bogus) then so does $P(Sx)$. It follows from the induction axiom that $\forall x P(x)$: there are no bogus numbers.[7]

6.2 A simple proof

Let's use the Peano axioms to prove something that we know to be true about the natural numbers we learned about in grade school but that might not be obvious from the axioms themselves. (This will give us some confidence that the axioms are not bogus.) We want to show that 0 is the only number that is not a successor:

Claim 1. $\forall x : (x \neq 0) \Rightarrow (\exists y : x = Sy)$.

To find a proof of this, we start by looking at the structure of what we are trying to prove. It's a universal statement about elements of $\mathbb{N}$ (implicitly, the $\forall x$ is really $\forall x \in \mathbb{N}$, since our axioms exclude anything that isn't in $\mathbb{N}$), so our table of proof techniques suggests using an induction argument, which in this case means finding some predicate we can plug into the induction schema.

[7] There is a complication here. Peano's original axioms were formulated in terms of second-order logic, which allows quantification over all possible predicates (you can write things like $\forall P P(x) \Rightarrow P(Sx)$). So the *bogus* predicate we defined is implicitly included in that for-all. In first-order logic, which is what everybody uses now, you can't quantify over predicates, and in particular if there is no predicate that distinguishes bogus numbers from legitimate ones, the induction axiom won't kick them out. This means that the Peano axioms (in their modern form) actually *do* allow bogus numbers to sneak in somewhere around infinity. But they have to be *very polite* bogus numbers that never do anything different from ordinary numbers. This is probably not a problem except for philosophers. Similar problems show up for any model with infinitely many elements, due to something called the Löwenheim-Skolem theorem.

If we strip off the $\forall x$, we are left with

$$(x \neq 0) \Rightarrow (\exists y : x = Sy).$$

Here a direct proof is suggested: assuming $x \neq 0$, and try to prove $\exists y : x = Sy$. But our axioms don't tell us much about numbers that aren't 0, so it's not clear what to do with the assumption. This turns out to be a dead end.

Recalling that $A \Rightarrow B$ is the same thing as $\neg A \vee B$, we can rewrite our goal as

$$x = 0 \vee \exists y : x = Sy.$$

This seems like a good candidate for P (our *induction hypothesis*), because we do know a few things about 0. Let's see what happens if we try plugging this into the induction schema:

- $P(0) \equiv 0 = 0 \vee \exists y : 0 = Sy$. The right-hand term looks false because of our first axiom, but the left-hand term is just the reflexive axiom for equality. $P(0)$ is true.
- $\forall x P(x) \Rightarrow P(Sx)$. We can drop the $\forall x$ if we fix an arbitrary x. Expand the right-hand side $P(Sx) \equiv Sx = 0 \vee \exists y Sx = Sy$. We can be pretty confident that $Sx \neq 0$ (it's an axiom), so if this is true, we had better show $\exists y Sx = Sy$. The first thing to try for $\exists$ statements is instantiation: pick a good value for y. Picking $y = x$ works.

Since we showed $P(0)$ and $\forall x P(x) \Rightarrow P(Sx)$, the induction schema tells us $\forall x P(x)$. This finishes the proof.

Having figured the proof out, we might go back and clean up any false starts to produce a compact version. A typical mathematician might write the preceding argument as:

Proof. By induction on x. For x = 0, the premise fails. For Sx, let $y = x$. □

A really lazy mathematician would write:

Proof. Induction on x. □

Though laziness is generally a virtue, you probably shouldn't be quite this lazy when writing up homework assignments.

6.3 Defining addition

Because of our restricted language, we do not yet have the ability to state valuable facts like $1 + 1 = 2$ (which we would have to write as $S0 + S0 = SS0$). Let's fix this, by adding a two-argument function symbol + which we will define using the axioms

- $x + 0 = x$.
- $x + Sy = S(x + y)$.

(We are omitting some $\forall$ quantifiers, since unbounded variables are implicitly universally quantified.)

This definition is essentially a recursive program for computing $x + y$ using only successor, and there are some programming languages (e.g. Haskell) that will allow you to define addition using almost exactly this notation. If the definition works for all inputs to $+$, we say that $+$ is **well-defined**. Not working would include giving different answers depending on which parts of the definitions we applied first, or giving no answer for some particular inputs. These bad outcomes correspond to writing a buggy program. Though we can in principle prove that this particular definition is well-defined (using induction on y), we won't bother. Instead, we will try to prove things about our new concept of *addition* that will, among other things, tell us that the definition gives the *correct* answers.

We start with a **lemma**, which is Greek for a result that is not especially useful by itself but is handy for proving other results.[8]

Lemma 1. $0 + x = x$.

Proof. By induction on x. When $x = 0$, we have $0+0 = 0$, which is true from the first case of the definition. Now suppose $0 + x = x$ and consider what happens with Sx. We want to show $0+Sx = Sx$. Rewrite $0+Sx$ as $S(0+x)$ [second case of the definition], and use the induction hypothesis to show $S(0+x) = S(x)$. □

(We could do a lot of QED-ish jumping around in the end zone there, but it is more refined—and *lazier*—to leave off the end of the proof once it's clear we've satisifed all of our obligations.)

Here's another lemma, which looks equally useless:

Lemma 2. $x + Sy = Sx + y$.

Proof. By induction on y. If $y = 0$, then $x + S0 = S(x + 0) = Sx = Sx + 0$. Now suppose the result holds for y and show $x + SSy = Sx + Sy$. We have $x + SSy = S(x + Sy) = S(Sx + y)$[ind. hyp.] $= Sx + Sy$. □

Now we can prove a **theorem**: this is a result that we think of as useful in its own right. (In programming terms, it's something we export from a module instead of hiding inside it as an internal procedure.)

Theorem 2. $x + y = y + x$. *(Commutativity of addition.)*

Proof. By induction on x. If $x = 0$, then $0 + y = y + 0$ (see previous lemma). Now suppose $x + y = y + x$, and we want to show $Sx + y = y + Sx$. But $y + Sx = S(y + x)$[axiom] $= S(x + y)$[induction hypothesis] $= x + Sy$[axiom] $= Sx + y$[lemma]. □

This sort of definition-lemma-lemma-theorem structure is typical of written mathematical proofs. Breaking things down into small pieces (just like breaking

[8] It really means *fork*.

big subroutines into small subroutines) makes debugging easier, since you can check if some intermediate lemma is true or false without having to look through the entire argument at once.

Question: How do you know which lemmas to prove? Answer: As when writing code, you start by trying to prove the main theorem, and whenever you come across something you need and can't prove immediately, you fork it off as a lemma. Conclusion: The preceding notes were not originally written in order.

6.3.1 Other useful properties of addition

So far we have shown that $x + y = y + x$, also known as **commutativity of addition**. Another familiar property is **associativity of addition**: $x + (y + z) = (x + y) + z$. This is easily proven by induction (try it!)

We don't have subtraction in $\mathbb{N}$ (what's $3 - 5$?)[9] The closest we can get is **cancellation**:

Lemma 3. $x + y = x + z \Rightarrow y = z$.

We can define $\leq$ for $\mathbb{N}$ directly from addition: Let $x \leq y \equiv \exists z x + z = y$. Then we can easily prove each of the following (possibly using our previous results about addition having commutativity, associativity, and cancellation).

- $0 \leq x$.
- $x \leq Sx$.
- $x \leq y \wedge y \leq z \Rightarrow x \leq z$.
- $a \leq b \wedge c \leq d \Rightarrow a + c \leq b + d$.
- $x \leq y \wedge y \leq x \Rightarrow x = y$.

(The actual proofs will be left as an exercise for the reader.)

6.4 A scary induction proof involving even numbers

Let's define the predicate $\text{Even}(x) \equiv \exists y x = y+y$. (The use of $\equiv$ here signals that $\text{Even}(x)$ is syntactic sugar, and we should think of any occurrence of $\text{Even}(x)$ as expanding to $\exists y x = y + y$.)

It's pretty easy to see that $0 = 0+0$ is even. Can we show that S0 is not even?

Lemma 4. $\neg \mathit{Even}(S0)$.

[9] This actually came up on a subtraction test I got in the first grade from the terrifying Mrs Garrison at Mountain Park Elementary School in Berkeley Heights, New Jersey. She insisted that -2 was not the correct answer, and that we should have recognized it as a trick question. She also made us black out the arrow the left of the zero on the number-line stickers we had all been given to put on the top of our desks. Mrs Garrison was, on the whole, a fine teacher, but she did not believe in New Math.

Proof. Expand the claim as $\neg\exists y S0 = y + y \equiv \forall y S0 \neq y + y$. Since we are working over $\mathbb{N}$, it's tempting to try to prove the $\forall y$ bit using induction. But it's not clear why $S0 \neq y + y$ would tell us anything about $S0 \neq Sy + Sy$. So instead we do a case analysis, using our earlier observation that every number is either 0 or Sz for some z.

Case 1 $y = 0$. Then $S0 \neq 0 + 0$ since $0 + 0 = 0$ (by the definition of $+$) and $0 \neq S0$ (by the first axiom).

Case 2 $y = Sz$. Then $y + y = Sz + Sz = S(Sz + z) = S(z + Sz) = SS(z + z)$.[10] Suppose $S0 = SS(z + z)$ [Note: "Suppose" usually means we are starting a proof by contradiction]. Then $0 = S(z + z)$ [second axiom], violating $\forall x 0 \neq Sx$ [first axiom]. So $S0 \neq SS(z + z) = y + y$.

Since we have $S0 \neq y + y$ in either case, it follows that $S0$ is not even. □

Maybe we can generalize this lemma! If we recall the pattern of non-even numbers we may have learned long ago, each of them $(1, 3, 5, 7, \dots)$ happens to be the successor of some even number $(0, 2, 4, 6, \dots)$. So maybe it holds that:

Theorem 3. *Even*$(x) \Rightarrow \neg$*Even*(Sx).

Proof. Expanding the definitions gives $(\exists y x = y + y) \Rightarrow (\neg\exists z Sx = z + z)$. This is an implication at top-level, which calls for a direct proof. The assumption we make is $\exists y x = y + y$. Let's pick some particular y that makes this true (in fact, there is only one, but we don't need this). Then we can rewrite the right-hand side as $\neg\exists z S(y + y) = z + z$. There doesn't seem to be any obvious way to show this (remember that we haven't invented subtraction or division yet, and we probably don't want to).

We are rescued by showing the stronger statement $\forall y \neg\exists z S(y + y) = z + z$: this is something we can prove by induction (on y, since that's the variable inside the non-disguised universal quantifier). Our previous lemma gives the base case $\neg\exists z S(0 + 0) = z + z$, so we just need to show that $\neg\exists z S(y + y) = z + z implies \neg\exists z S(Sy + Sy) = z + z$. Suppose that $S(Sy + Sy) = z + z$ for some z ["suppose" = proof by contradiction again: we are going to drive this assumption into a ditch]. Rewrite $S(Sy + Sy)$ to get $SSS(y + y) = z + z$. Now consider two cases:

Case 1 $z = 0$. Then $SSS(y + y) = 0 + 0 = 0$, contradicting our first axiom.

Case 2 $z = Sw$. Then $SSS(y + y) = Sw + Sw = SS(w + w)$. Applying the second axiom twice gives $S(y + y) = w + w$. But this contradicts the induction hypothesis.

Since both cases fail, our assumption must have been false. It follows that $S(Sy + Sy)$ is not even, and the induction goes through. □

[10] What justifies that middle step?

6.5 Defining more operations

Let's define multiplication ($\cdot$) by the axioms:

- $0 \cdot y = 0$.
- $Sx \cdot y = y + x \cdot y$.

Some properties of multiplication:

- $x \cdot 0 = 0$.
- $1 \cdot x = x$.
- $x \cdot 1 = x$.
- $x \cdot y = y \cdot x$.
- $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
- $x \neq 0 \wedge x \cdot y = x \cdot z \Rightarrow y = z$.
- $x \cdot (y + z) = x \cdot y + x \cdot z$.
- $x \leq y \Rightarrow z \cdot x \leq z \cdot y$.
- $z \neq 0 \wedge z \cdot x \leq z \cdot y \Rightarrow x \leq y$.

(Note we are using 1 as an abbreviation for $S0$.)

The first few of these are all proved pretty much the same way as for addition. Note that we can't divide in $\mathbb{N}$ any more than we can subtract, which is why we have to be content with multiplicative cancellation.

Exercise: Show that the $\text{Even}(x)$ predicate, defined previously as $\exists y y = x + x$, is equivalent to $\text{Even}'(x) \equiv \exists y x = 2 \cdot y$, where $2 = SS0$. Does this definition make it easier or harder to prove $\neg\text{Even}'(S0)$?