Jeder Genießende meint, dem Baume habe es an der Frucht gelegen;
aber ihm lag am Samen.
[Everyone who enjoys thinks that the fundamental thing about trees is the fruit,
but in fact it is the seed.]

— Friedrich Wilhelm Nietzsche,
Vermischte Meinungen und Sprüche [Mixed Opinions and Maxims] (1879)

In view of all the deadly computer viruses that have been spreading lately,
Weekend Update would like to remind you:
When you link up to another computer,
you're linking up to every computer that that computer has ever linked up to.

— Dennis Miller, "Saturday Night Live", (c. 1985)

Anything that, in happening, causes itself to happen again, happens again.

— Douglas Adams (2005)

Proof by Induction

Induction is a method for proving universally quantified propositions—statements about *all* elements of a (usually infinite) set. Induction is also the single most useful tool for reasoning about, developing, and analyzing algorithms. These notes give several examples of inductive proofs, along with a standard boilerplate and some motivation to justify (and help you remember) why induction works.

1 Prime Divisors: Proof by Smallest Counterexample

A ***divisor*** of a positive integer n is a positive integer p such that the ratio n/p is an integer. The integer 1 is a divisor of every positive integer (because $n/1 = n$), and every integer is a divisor of itself (because $n/n = 1$). A ***proper divisor*** of n is any divisor of n other than n itself. A positive integer is ***prime*** if it has *exactly two* divisors, which must be 1 and itself; equivalently; a number is prime if and only if 1 is its only proper divisor. A positive integer is ***composite*** if it has more than two divisors (or equivalently, more than one proper divisor). The integer 1 is neither prime nor composite, because it has exactly one divisor, namely itself.

Let's prove our first theorem:

Theorem 1. *Every integer greater than* 1 *has a prime divisor.*

The very first thing that you should notice, after reading just one word of the theorem, is that this theorem is *universally quantified*—it's a statement about *all* the elements of a set, namely, the set of positive integers larger than 1. If we were forced at gunpoint to write this sentence using fancy logic notation, the first character would be the universal quantifier ∀, pronounced 'for all'. Fortunately, that won't be necessary.

There are only two ways to prove a universally quantified statement: directly or by contradiction. Let's say that again, louder: ***There are only two ways to prove a universally quantified statement: directly or by contradiction.*** Here are the standard templates for these two methods, applied to Theorem 1:

> **Direct proof:** Let n be an arbitrary integer greater than 1.
>
> *. . . blah blah blah . . .*
>
> Thus, n has at least one prime divisor. □

> **Proof by contradiction:** For the sake of argument, assume there is an integer greater than 1 with no prime divisor.
> Let n be an arbitrary integer greater than 1 with no prime divisor.
> *. . . blah blah blah . . .*
> But that's just silly. Our assumption must be incorrect. □

The shaded boxes *. . . blah blah blah . . .* indicate missing proof details (which you have to fill in). Most people find proofs by contradiction easier to discover than direct proofs, so let's try that first.

> **Proof by contradiction:** For the sake of argument, assume there is an integer greater than 1 with no prime divisor.
> Let n be an arbitrary integer greater than 1 with no prime divisor.
> Since n is a divisor of n, and n has no prime divisors, n cannot be prime.
> Thus, n must have at least one divisor d such that $1 < d < n$.
> Let d be an arbitrary divisor of n such that $1 < d < n$.
> Since n has no prime divisors, d cannot be prime.
> Thus, d has at least one divisor d' such that $1 < d' < d$.
> Let d' be an arbitrary divisor of d such that $1 < d' < d$.
> Because d/d' is an integer, $n/d' = (n/d) \cdot (d/d')$ is also an integer.
> Thus, d' is also a divisor of n.
> Since n has no prime divisors, d' cannot be prime.
> Thus, d' has at least one divisor d'' such that $1 < d'' < d'$.
> Let d'' be an arbitrary divisor of d' such that $1 < d'' < d'$.
> Because d'/d'' is an integer, $n/d'' = (n/d') \cdot (d'/d'')$ is also an integer.
> Thus, d'' is also a divisor of n.
> Since n has no prime divisors, d'' cannot be prime.
> *. . . blah HELP! blah I'M STUCK IN AN INFINITE LOOP! blah . . .*
> But that's just silly. Our assumption must be incorrect. □

We seem to be stuck in an infinite loop, looking at smaller and smaller divisors $d > d' > d'' > \cdots$, none of which are prime. But this loop can't really be infinite. There are only $n - 1$ positive integers smaller than n, so the proof *must* end after *at most* $n-1$ iterations. But how do we turn this observation into a formal proof? We need a single, self-contained proof for all integers n; we're not allowed to write longer proofs for bigger integers. The trick is to jump directly to the ***smallest*** counterexample.

> **Proof by smallest counterexample:** For the sake of argument, assume that there is an integer greater than 1 with no prime divisor.
> Let n be ***the smallest*** integer greater than 1 with no prime divisor.
> Since n is a divisor of n, and n has no prime divisors, n cannot be prime.
> Thus, n has a divisor d such that $1 < d < n$.
> Let d be a divisor of n such that $1 < d < n$.
> ***Because n is the smallest counterexample***, d has a prime divisor.
> Let p be a prime divisor of d.
> Because d/p is an integer, $n/p = (n/d) \cdot (d/p)$ is also an integer.
> Thus, p is also a divisor of n.
> But this contradicts our assumption that n has no prime divisors!
> So our assumption must be incorrect. □

Hooray, our first proof! We're done!

Um. . . well. . . no, we're definitely *not* done. That's a first draft up there, not a final polished proof. We don't write proofs just to convince ourselves; proofs are primarily a tool to convince other people. (In particular, 'other people' includes the people grading your homeworks and exams.) And while proofs by contradiction are usually easier to *write*, direct proofs are almost always easier to *read*. So as a service to our audience (and our grade), let's transform our minimal-counterexample proof into a direct proof.

Let's first rewrite the indirect proof slightly, to make the structure more apparent. First, we break the assumption that n is the smallest counterexample into three simpler assumptions: (1) n is an integer greater than 1; (2) n has no prime divisors; and (3) there are no smaller counterexamples. Second, instead of dismissing the possibility than n is prime out of hand, we include an explicit case analysis.

> **Proof by smallest counterexample:** Let n be an arbitrary integer greater than 1.
> For the sake of argument, suppose n has no prime divisor.
> ***Assume that every integer k such that $1 < k < n$ has a prime divisor.***
> There are two cases to consider: Either n is prime, or n is composite.
> - Suppose n is prime.
>   Then n is a prime divisor of n.
> - Suppose n is composite.
>   Then n has a divisor d such that $1 < d < n$.
>   Let d be a divisor of n such that $1 < d < n$.
>   ***Because no counterexample is smaller than n***, d has a prime divisor.
>   Let p be a prime divisor of d.
>   Because d/p is an integer, $n/p = (n/d) \cdot (d/p)$ is also an integer.
>   Thus, p is a prime divisor of n.
>
> In each case, we conclude that n has a prime divisor.
> But this contradicts our assumption that n has no prime divisors!
> So our assumption must be incorrect. □

Now let's look carefully at the structure of this proof. First, we assumed that the statement we want to prove is false. Second, we proved that the statement we want to prove is true. Finally, we concluded from the contradiction that our assumption that the statement we want to prove is false is incorrect, so the statement we want to prove must be true.

But that's just silly. Why do we need the first and third steps? After all, the second step is a proof all by itself! Unfortunately, this redundant style of proof by contradiction is *extremely* common, even in professional papers. Fortunately, it's also very easy to avoid; just remove the first and third steps!

> **Proof by induction:** Let n be an arbitrary integer greater than 1.
> ***Assume that every integer k such that $1 < k < n$ has a prime divisor.***
> There are two cases to consider: Either n is prime or n is composite.
> - First, suppose n is prime.
>   Then n is a prime divisor of n.
> - Now suppose n is composite.
>   Then n has a divisor d such that $1 < d < n$.
>   Let d be a divisor of n such that $1 < d < n$.
>   ***Because no counterexample is smaller than n***, d has a prime divisor.
>   Let p be a prime divisor of d.
>   Because d/p is an integer, $n/p = (n/d) \cdot (d/p)$ is also an integer.
>   Thus, p is a prime divisor of n.
>
> In both cases, we conclude that n has a prime divisor. □

This style of proof is called ***induction***.[1] The assumption that there are no counterexamples smaller than n is called the ***induction hypothesis***. The two cases of the proof have different names. The

[1] Many authors use the high-falutin' name *the principle of mathematical induction*, to distinguish it from *inductive reasoning*, the informal process by which we conclude that pigs can't whistle, horses can't fly, and NP-hard problems cannot be solved in polynomial time. We already know that every proof is mathematical (and arguably, all mathematics is proof), so as a description of a *proof* technique, the adjective 'mathematical' is simply redundant.

first case, which we argue directly, is called the ***base case***. The second case, which actually uses the induction hypothesis, is called the ***inductive case***. You may find it helpful to actually label the induction hypothesis, the base case(s), and the inductive case(s) in your proof.

The following point cannot be emphasized enough: The only difference between a proof by induction and a proof by smallest counterexample is the way we write down the argument. The essential structure of the proofs are exactly the same. The core of our original indirect argument is a proof of the following implication for all n:

$$n \text{ has no prime divisor} \implies \text{some number smaller than } n \text{ has no prime divisor.}$$

The core of our direct proof is the following logically equivalent implication:

$$\text{every number smaller than } n \text{ has a prime divisor} \implies n \text{ has a prime divisor}$$

The left side of this implication is just the induction hypothesis.

The proofs we've been playing with have been very careful and explicit; until you're comfortable writing your own proofs, you should be equally careful. A more mature proof-writer might express the same proof more succinctly as follows:

> **Proof by induction:** Let n be an arbitrary integer greater than 1. Assume that every integer k such that $1 < k < n$ has a prime divisor. If n is prime, then n is a prime divisor of n. On the other hand, if n is composite, then n has a proper divisor d. The induction hypothesis implies that d has a prime divisor p. The integer p is also a divisor of n. □

A proof in this more succinct form is still worth full credit, provided the induction hypothesis is written explicitly and the case analysis is obviously exhaustive.

A professional mathematician would write the proof even more tersely:

> **Proof:** Induction. □

And you can write that tersely, too, when you're a professional mathematician.

2 The Axiom of Induction

Why does this work? Well, let's step back to the original proof by smallest counterexample. How do we know that a smallest counterexample exists? This seems rather obvious, but in fact, it's impossible to prove without using the following fact:

Every non-empty set of positive integers has a smallest element.

Every set X of positive integers is the set of counterexamples to some proposition $P(n)$ (specifically, the proposition $n \notin X$). Thus, the Axiom of Induction can be rewritten as follows:

If the proposition $P(n)$ is false for some positive integer n,
then
the proposition $(P(1) \wedge P(2) \wedge \cdots \wedge P(n-1) \wedge \neg P(n))$ is true for some positive integer n.

Equivalently, in English:

If some statement about positive integers has a counterexample,
then
that statement has a smallest counterexample.

We can write this implication in contrapositive form as follows:

> *If the proposition $(P(1) \wedge P(2) \wedge \cdots \wedge P(n-1) \wedge \neg P(n))$ is false for every positive integer n,*
> *then*
> *the proposition $P(n)$ is true for every positive integer n.*

Finally, let's rewrite the first half of this statement in a logically equivalent form, by replacing $\neg(p \wedge \neg q)$ with $p \rightarrow q$.

> *If the implication $(P(1) \wedge P(2) \wedge \cdots \wedge P(n-1)) \rightarrow P(n)$ is true for every positive integer n,*
> *then*
> *the proposition $P(n)$ is true for every positive integer n.*

This formulation is usually called the ***Axiom of Induction***. In a proof by induction that $P(n)$ holds for all n, the conjunction $(P(1) \wedge P(2) \wedge \cdots \wedge P(n-1))$ is the inductive hypothesis.

A ***proof by induction*** for the proposition "$P(n)$ for every positive integer n" is nothing but a direct proof of the more complex proposition "$(P(1) \wedge P(2) \wedge \cdots \wedge P(n-1)) \rightarrow P(n)$ for every positive integer n". Because it's a direct proof, it *must* start by considering an arbitrary positive integer, which we might as well call n. Then, to prove the implication, we explicitly assume the hypothesis $(P(1) \wedge P(2) \wedge \cdots \wedge P(n-1))$ and then prove the conclusion $P(n)$ *for that particular value of* n. The proof almost always breaks down into two or more cases, each of which may or may not actually use the inductive hypothesis.

Here is the boilerplate for *every* induction proof. Read it. Learn it. Use it.

> **Theorem:** $P(n)$ for every positive integer n.
>
> **Proof by induction:** Let n be an arbitrary positive integer.
> Assume inductively that $P(k)$ is true for every positive integer $k < n$.
> There are several cases to consider:
> - Suppose n is *. . . blah blah blah . . .*
>   Then $P(n)$ is true.
> - Suppose n is *. . . blah blah blah . . .*
>   The inductive hypothesis implies that *. . . blah blah blah . . .*
>   Thus, $P(n)$ is true.
>
> In each case, we conclude that $P(n)$ is true. □

Some textbooks distinguish between several different types of induction: 'regular' induction versus 'strong' induction versus 'complete' induction versus 'structural' induction versus 'transfinite' induction. These distinctions are utterly pointless; we will not even define them. Every 'different type' of induction proof is provably equivalent to a proof by smallest counterexample. (Later we will consider inductive proofs of statements about partially ordered sets other than the positive integers, for which 'smallest' has a different meaning, but this difference will prove to be inconsequential.)

3 Stamps and Recursion

Let's move on to a completely different example.

Theorem 2. *Given an unlimited supply of 5-cent stamps and 7-cent stamps, we can make any amount of postage larger than 23 cents.*

We could prove this by contradiction, using a smallest-counterexample argument, but let's aim for a direct proof by induction this time. We start by writing down the induction boilerplate, using the standard induction hypothesis: *There is no counterexample smaller than n.*

> **Proof by induction:** Let n be an arbitrary integer greater than 23.
> Assume that for any integer k such that $23 < k < n$, we can make k cents in postage.
> *. . . blah blah blah . . .*
> Thus, we can make n cents in postage. □

How do we fill in the details? One approach is to think about what you would actually do if you really had to make n cents in postage. For example, you might start with a 5-cent stamp, and then try to make $n-5$ cents in postage. The inductive hypothesis says you can make *any* amount of postage bigger than 23 cents and less than n cents. So if $n-5 > 23$, then *you already know* that you can make $n-5$ cents in postage! (You don't know *how* to make $n-5$ cents in postage, but so what?)

Let's write this observation into our proof as two separate cases: either $n \geq 30$ (where our approach works) and $n \leq 29$ (where we don't know what to do yet).

> **Proof by induction:** Let n be an arbitrary integer greater than 23.
> Assume that for any integer k such that $23 < k < n$, we can make k cents in postage.
> There are two cases to consider: Either $n > 28$ or $n \leq 28$.
> - Suppose $n > 28$.
>   Then $23 < n - 5 < n$.
>   Thus, ***the induction hypothesis*** implies that we can make $n-5$ cents in postage.
>   Adding one more 5-cent stamp gives us n cents in postage.
> - Now suppose $n \leq 28$.
>
> *. . . blah blah blah . . .*
> In both cases, we can make n cents in postage. □

What do we do in the second case? Fortunately, this case considers only five integers: 24, 25, 26, 27, and 28. There might be a clever way to solve all five cases at once, but why bother? They're small enough that we can find a solution by brute force in less than a minute. To make the proof more readable, I'll unfold the nested cases and list them in increasing order.

> **Proof by induction:** Let n be an arbitrary integer greater than 23.
> Assume that for any integer k such that $23 < k < n$, we can make k cents in postage.
> There are six cases to consider: $n = 24$, $n = 25$, $n = 26$, $n = 27$, $n = 28$, and $n > 28$.
> - $24 = 7 + 7 + 5 + 5$
> - $25 = 5 + 5 + 5 + 5 + 5$
> - $26 = 7 + 7 + 7 + 5$
> - $27 = 7 + 5 + 5 + 5 + 5$
> - $28 = 7 + 7 + 7 + 7$
> - Suppose $n > 28$.
>   Then $23 < n - 5 < n$.
>   Thus, ***the induction hypothesis*** implies that we can make $n-5$ cents in postage.
>   Adding one more 5-cent stamp gives us n cents in postage.
>
> In all cases, we can make n cents in postage. □

Voilà! An induction proof! More importantly, we now have a recipe for *discovering* induction proofs.

1. ***Write down the boilerplate.*** Write down the universal invocation ('Let n be an arbitrary. . . '), the induction hypothesis, and the conclusion, with enough blank space for the remaining details. Don't be clever. Don't even think. Just write. ***This is the easy part.*** To emphasize the common structure, the boilerplate will be indicated in green for the rest of this handout.

2. ***Think big.*** *Don't* think how to solve the problem all the way down to the ground; you'll only make yourself dizzy. *Don't* think about piddly little numbers like 1 or 5 or 10^{100}. Instead, think about how to reduce the proof about some *absfoluckingutely ginormous* value of n to a proof about some other number(s) smaller than n. ***This is the hard part.***

3. ***Look for holes.*** Look for cases where your inductive argument breaks down. Solve those cases directly. Don't be clever here; be stupid but thorough.

4. ***Rewrite everything.*** Your first proof is a rough draft. Rewrite the proof so that your argument is easier for your (unknown?) reader to follow.

The cases in an inductive proof always fall into two categories. Any case that uses the inductive hypothesis is called an *inductive case*. Any case that does not use the inductive hypothesis is called a *base case*. Typically, but *not* always, base cases consider a few small values of n, and the inductive cases consider everything else. Induction proofs are usually clearer if we present the base cases first, but I find it much easier to *discover* the inductive cases first. In other words, I recommend writing induction proofs backwards.

Well-written induction proofs *very* closely resemble well-written recursive programs. We computer scientists use induction primarily to reason about recursion, so maintaining this resemblance is extremely useful—we only have to keep one mental pattern, called 'induction' when we're writing proofs and 'recursion' when we're writing code. Consider the following C and Scheme programs for making n cents in postage:

```
void postage(int n)
{
    assert(n>23);
    switch ($n$)
    {
        case 24:  printf("7+7+5+5");    break;
        case 25:  printf("5+5+5+5+5");  break;
        case 26:  printf("7+7+7+5");    break;
        case 27:  printf("7+5+5+5+5");  break;
        case 28:  printf("7+7+7+7");    break;
        default:
            postage(n-5);
            printf("+5");
    }
}
```

```
(define (postage n)
  (cond ((= n 24) (5 5 7 7))
        ((= n 25) (5 5 5 5 5))
        ((= n 26) (5 7 7 7))
        ((= n 27) (5 5 5 5 7))
        ((= n 28) (7 7 7 7))
        ((> n 28) (cons 5 (postage (- n 5))))))
```

The C program begins by declaring the input parameter ("Let n be an arbitrary integer. . . ") and asserting its range (". . . greater than 23."). (Scheme programs don't have type declarations.) In both languages, the code branches into six cases: five that are solved directly, plus one that is handled ~~by invoking the inductive hypothesis~~ recursively.

4 More on Prime Divisors

Before we move on to different examples, let's prove another fact about prime numbers:

Theorem 3. *Every positive integer is a product of prime numbers.*

First, let's write down the boilerplate. Hey! I saw that! You were *thinking*, weren't you? Stop that this instant! Don't make me turn the car around. ***First*** we write down the boilerplate.

> **Proof by induction:** Let n be an arbitrary positive integer.
> Assume that any positive integer $k < n$ is a product of prime numbers.
> There are *some* cases to consider:
>
> *. . . blah blah blah . . .*
>
> Thus, n is a product of prime numbers. □

Now let's think about how you would actually factor a positive integer n into primes. There are a couple of different options here. One possibility is to find a prime divisor p of n, as guaranteed by Theorem 1, and recursively factor the integer n/p. This argument works as long as $n \geq 2$, but what about $n = 1$? The answer is simple: 1 *is the product of the* ***empty*** *set of primes*. What else could it be?

> **Proof by induction:** Let n be an arbitrary positive integer.
> Assume that any positive integer $k < n$ is a product of prime numbers.
> There are two cases to consider: either $n = 1$ or $n \geq 2$.
>
> - If $n = 1$, then n is the product of the elements of the empty set, each of which is prime, green, sparkly, vanilla, and hemophagic.
> - Suppose $n > 1$. Let p be a prime divisor of n, as guaranteed by Theorem 2. The inductive hypothesis implies that the positive integer n/p is a product of primes, and clearly $n = (n/p) \cdot p$.
>
> In both cases, n is a product of prime numbers. □

But an even simpler method is to factor n into any two proper divisors, and recursively handle them both. This method works as long as n is composite, since otherwise there is no way to factor n into smaller integers. Thus, we need to consider prime numbers separately, as well as the special case 1.

> **Proof by induction:** Let n be an arbitrary positive integer.
> Assume that any positive integer $k < n$ is a product of prime numbers.
> There are three cases to consider: either $n = 1$, n is prime, or n is composite.
>
> - If $n = 1$, then n is the product of the elements of the empty set, each of which is prime, red, broody, chocolate, and lycanthropic.
> - If n is prime, then n is the product of one prime number, namely n.
> - Suppose n is composite. Let d be any proper divisor of n (guaranteed by the definition of 'composite'), and let $m = n/d$. Since both d and m are positive integers smaller than n, the inductive hypothesis implies that d and m are both products of prime numbers. We clearly have $n = d \cdot m$.
>
> In both cases, n is a product of prime numbers. □

5 Summations

Here's an easy one.

Theorem 4. $\sum_{i=0}^{n} 3^i = \frac{3^{n+1}-1}{2}$ *for every non-negative integer n.*

First let's write down the induction boilerplate, which empty space for the details we'll fill in later.

> **Proof by induction:** Let n be an arbitrary non-negative integer.
>
> Assume inductively that $\sum_{i=0}^{k} 3^i = \frac{3^{k+1}-1}{2}$ for every non-negative integer $k < n$.
>
> There are *some number of* cases to consider:
>
> *... blah blah blah ...*
>
> We conclude that $\sum_{i=0}^{n} 3^i = \frac{3^{n+1}-1}{2}$. □

Now imagine you are part of an infinitely long assembly line of mathematical provers, each assigned to a particular non-negative integer. Your task is to prove this theorem for the integer 8675310. The regulations of the Mathematical Provers Union require you not to think about any other integer but your own. The assembly line starts with the Senior Master Prover, who proves the theorem for the case $n = 0$. Next is the Assistant Senior Master Prover, who proves the theorem for $n = 1$. After him is the Assistant Assistant Senior Master Prover, who proves the theorem for $n = 2$. Then the Assistant Assistant Assistant Senior Master Prover proves the theorem for $n = 3$. As the work proceeds, you start to get more and more bored. You attempt strike up a conversation with Jenny, the prover to your left, but she ignores you, preferring to focus on the proof. Eventually, you fall into a deep, dreamless sleep. An undetermined time later, Jenny wakes you up by shouting, "Hey, doofus! It's your turn!" As you look around, bleary-eyed, you realize that Jenny and everyone to your left has finished their proofs, and that everyone is waiting for you to finish yours. What do you do?

What you do, after wiping the drool off your chin, is stop and think for a moment about what you're trying to prove. What does that $\sum$ notation actually mean? Intuitively, we can expand the notation as follows:

$$\sum_{i=0}^{8675310} 3^i = 3^0 + 3^1 + \cdots + 3^{8675309} + 3^{8675310}.$$

Notice that this expression also contains the summation that Jenny just finished proving something about:

$$\sum_{i=0}^{8675309} 3^i = 3^0 + 3^1 + \cdots + 3^{8675308} + 3^{8675309}.$$

Putting these two expressions together gives us the following identity:

$$\sum_{i=0}^{8675310} 3^i = \sum_{i=0}^{8675309} 3^i \; + \; 3^{8675310}$$

In fact, this recursive identity is the *definition* of $\sum$. Jenny just proved that the summation on the right is equal to $(3^{8675310} - 1)/2$, so we can plug that into the right side of our equation:

$$\sum_{i=0}^{8675310} 3^i = \sum_{i=0}^{8675309} 3^i \; + \; 3^{8675310} = \frac{3^{8675310}-1}{2} + 3^{8675310}.$$

And it's all downhill from here. After a little bit of algebra, you simplify the right side of this equation to $(3^{8675311} - 1)/2$, wake up the prover to your right, and start planning your well-earned vacation.

Let's insert this argument into our boilerplate, only using a generic 'big' integer n instead of the specific integer 8675310:

> **Proof by induction:** Let n be an arbitrary non-negative integer.
>
> Assume inductively that $\sum_{i=0}^{k} 3^i = \frac{3^{k+1}-1}{2}$ for every non-negative integer $k < n$.
>
> There are two cases to consider: Either n *is big* or n *is small*.
>
> - If n *is big*, then
>
> $$\begin{aligned} \sum_{i=0}^{n} 3^i &= \sum_{i=0}^{n-1} 3^i + 3^n && \text{[definition of } \textstyle\sum\text{]} \\ &= \frac{3^n - 1}{2} + 3^n && \text{[induction hypothesis, with } k = n-1\text{]} \\ &= \frac{3^{n+1}-1}{2} && \text{[algebra]} \end{aligned}$$
>
> - On the other hand, if n *is small*, then *. . . blah blah blah . . .*
>
> In both cases, we conclude that $\sum_{i=0}^{n} 3^i = \frac{3^{n+1}-1}{2}$. □

Now, how big is 'big', and what do we do when n is 'small'? To answer the first question, let's look at where our existing inductive argument breaks down. In order to apply the induction hypothesis when $k = n-1$, the integer $n-1$ must be non-negative; equivalently, n must be *positive*. But that's the only assumption we need: ***The only case we missed is $\mathbf{n = 0}$.*** Fortunately, this case is easy to handle directly.

> **Proof by induction:** Let n be an arbitrary non-negative integer.
>
> Assume inductively that $\sum_{i=0}^{k} 3^i = \frac{3^{k+1}-1}{2}$ for every non-negative integer $k < n$.
>
> There are two cases to consider: Either $n = 0$ or $n \geq 1$.
>
> - If $n = 0$, then $\sum_{i=0}^{n} 3^i = 3^0 = 1$, and $\frac{3^{n+1}-1}{2} = \frac{3^1-1}{2} = 1$.
> - On the other hand, if $n \geq 1$, then
>
> $$\begin{aligned} \sum_{i=0}^{n} 3^i &= \sum_{i=0}^{n-1} 3^i + 3^n && \text{[definition of } \textstyle\sum\text{]} \\ &= \frac{3^n - 1}{2} + 3^n && \text{[induction hypothesis, with } k = n-1\text{]} \\ &= \frac{3^{n+1}-1}{2} && \text{[algebra]} \end{aligned}$$
>
> In both cases, we conclude that $\sum_{i=0}^{n} 3^i = \frac{3^{n+1}-1}{2}$. □

Here is the same proof, written more tersely; the non-standard symbol $\overset{IH}{=\!=}$ indicates the use of the induction hypothesis.

Proof by induction: Let n be an arbitrary non-negative integer, and assume inductively that $\sum_{i=0}^{k} 3^i = (3^{k+1} - 1)/2$ for every non-negative integer $k < n$. The base case $n = 0$ is trivial, and for any $n \geq 1$, we have

$$\sum_{i=0}^{n} 3^i = \sum_{i=0}^{n-1} 3^i + 3^n \overset{IH}{=\!=} \frac{3^n - 1}{2} + 3^n = \frac{3^{n+1} - 1}{2}.$$

□

This is not the only way to prove this theorem by induction; here is another:

Proof by induction: Let n be an arbitrary non-negative integer, and assume inductively that $\sum_{i=0}^{k} 3^i = (3^{k+1} - 1)/2$ for every non-negative integer $k < n$. The base case $n = 0$ is trivial, and for any $n \geq 1$, we have

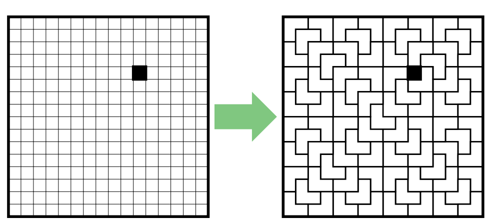
$$\sum_{i=0}^{n} 3^i = 3^0 + \sum_{i=1}^{n} 3^i = 3^0 + 3 \cdot \sum_{i=0}^{n-1} 3^i \overset{IH}{=\!=} 3^0 + 3 \cdot \frac{3^n - 1}{2} = \frac{3^{n+1} - 1}{2}.$$

□

In the remainder of these notes, I'll give several more examples of induction proofs. In some cases, I give multiple proofs for the same theorem. Unlike the earlier examples, I will not describe the thought process that lead to the proof; in each case, I followed the basic outline on page 7.

6 Tiling with Triominos

The next theorem is about *tiling* a square checkerboard with *triominos*. A triomino is a shape composed of three squares meeting in an L-shape. Our goal is to cover as much of a $2^n \times 2^n$ grid with triominos as possible, without any two triominos overlapping, and with all triominos inside the square. We can't cover every square in the grid—the number of squares is 4^n, which is not a multiple of 3—but we can cover all but one square. In fact, as the next theorem shows, we can choose *any* square to be the one we don't want to cover.



Almost tiling a 16×16 checkerboard with triominos.

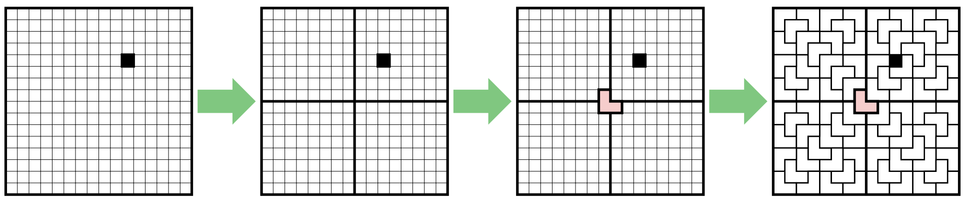
Theorem 5. *For any non-negative integer n, the $2^n \times 2^n$ checkerboard with any square removed can be tiled using L-shaped triominos.*

Here are two different inductive proofs for this theorem, one 'top down', the other 'bottom up'.

Proof by top-down induction: Let n be an arbitrary non-negative integer. Assume that for any non-negative integer $k < n$, the $2^k \times 2^k$ grid with any square removed can be tiled using triominos. There are two cases to consider: Either $n = 0$ or $n \geq 1$.

- The $2^0 \times 2^0$ grid has a single square, so removing one square leaves nothing, which we can tile with zero triominos.
- Suppose $n \geq 1$. In this case, the $2^n \times 2^n$ grid can be divided into four smaller $2^{n-1} \times 2^{n-1}$ grids. Without loss of generality, suppose the deleted square is in the upper right quarter. With a single L-shaped triomino at the center of the board, we can cover one square in each of the other three quadrants. The induction hypothesis implies that we can tile each of the quadrants, minus one square.

In both cases, we conclude that the $2^n \times 2^n$ grid with any square removed can be tiled with triominos. □

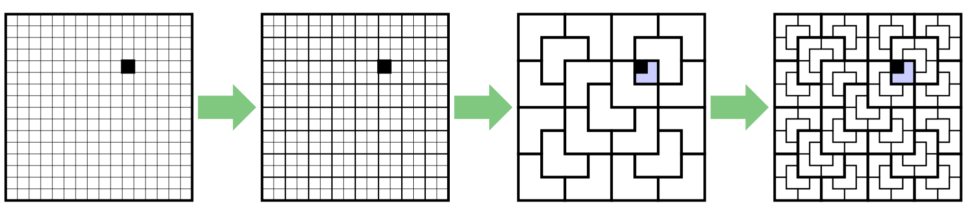


Top-down inductive proof of Theorem 4.

Proof by bottom-up induction: Let n be an arbitrary non-negative integer. Assume that for any non-negative integer $k < n$, the $2^k \times 2^k$ grid with any square removed can be tiled using triominos. There are two cases to consider: Either $n = 0$ or $n \geq 1$.

- The $2^0 \times 2^0$ grid has a single square, so removing one square leaves nothing, which we can tile with zero triominos.
- Suppose $n \geq 1$. Then by clustering the squares into 2×2 *blocks*, we can transform any $2^n \times 2^n$ grid into a $2^{n-1} \times 2^{n-1}$ grid. Suppose square (i, j) has been removed from the $2^n \times 2^n$ grid. The induction hypothesis implies that the $2^{n-1} \times 2^{n-1}$ grid with block $(\lfloor i/2 \rfloor, \lfloor j/2 \rfloor)$ removed can be tiled with double-size triominos. Each double-size triomono can be tiled with four smaller triominos, and block $(\lfloor i/2 \rfloor, \lfloor j/2 \rfloor)$ with square (i, j) removed is another triomino.

In both cases, we conclude that the $2^n \times 2^n$ grid with any square removed can be tiled with triominos. □



Second proof of Theorem 4.

7 Binary Numbers Exist

Theorem 6. *Every non-negative integer can be written as the sum of distinct powers of 2.*

Intuitively, this theorem states that every number can be represented in binary. (That's not a proof, by the way; it's just a restatement of the theorem.) I'll present *four* distinct inductive proofs for this theorem. The first two are standard, by-the-book induction proofs.

Proof by top-down induction: Let n be an arbitrary non-negative integer. Assume that any non-negative integer less than n can be written as the sum of distinct powers of 2. There are two cases to consider: Either $n = 0$ or $n \geq 1$.

- The base case $n = 0$ is trivial—the elements of the empty set are distinct and sum to zero.

- Suppose $n \geq 1$. Let k be the largest integer such that $2^k \leq n$, and let $m = n - 2^k$. Observe that $m < 2^{k+1} - 2^k = 2^k$. Because $0 \leq m < n$, the inductive hypothesis implies that m can be written as the sum of distinct powers of 2. Moreover, in the summation for m, each power of 2 is at most m, and therefore less than 2^k. Thus, $m + 2^k$ is the sum of distinct powers of 2.

In either case, we conclude that n can be written as the sum of distinct powers of 2. □

Proof by bottom-up induction: Let n be an arbitrary non-negative integer. Assume that any non-negative integer less than n can be written as the sum of distinct powers of 2. There are two cases to consider: Either $n = 0$ or $n \geq 1$.

- The base case $n = 0$ is trivial—the elements of the empty set are distinct and sum to zero.

- Suppose $n \geq 1$, and let $m = \lfloor n/2 \rfloor$. Because $0 \leq m < n$, the inductive hypothesis implies that m can be written as the sum of distinct powers of 2. Thus, $2m$ can also be written as the sum of distinct powers of 2, each of which is greater than 2^0. If n is even, then $n = 2m$ and we are done; otherwise, $n = 2m + 2^0$ is the the sum of distinct powers of 2.

In either case, we conclude that n can be written as the sum of distinct powers of 2. □

The third proof deviates slightly from the induction boilerplate. At the top level, this proof doesn't actually use induction at all! However, a key step requires its own (straightforward) inductive proof.

Proof by algorithm: Let n be an arbitrary non-negative integer. Let S be a multiset containing n copies of 2^0. Modify S by running the following algorithm:

```
while S has more than one copy of any element 2^i
    Remove two copies of 2^i from S
    Insert one copy of 2^(i+1) into S
```

Each iteration of this algorithm reduces the cardinality of S by 1, so the algorithm must eventually halt. When the algorithm halts, the elements of S are distinct. We claim that just after each iteration of the while loop, the elements of S sum to n.

> **Proof by induction:** Consider an arbitrary iteration of the loop. Assume inductively that just after each previous iteration, the elements of S sum to n. Before any iterations of the loop, the elements of S sum to n by definition. The induction hypothesis implies that just before the current iteration begins, the elements of S sum to n. The loop replaces two copies of some number 2^i with their sum 2^{i+1}, leaving the total sum of S unchanged. Thus, when the iteration ends, the elements of S sum to n. □

Thus, when the algorithm halts, the elements of S are distinct powers of 2 that sum to n. We conclude that n can be written as the sum of distinct powers of 2. □

The fourth proof uses so-called 'weak' induction, where the inductive hypothesis can only be applied at $n-1$. Not surprisingly, tying all but one hand behind our backs makes the resulting proof longer, more complicated, and harder to read. It doesn't help that the algorithm used in the proof is overly specific. Nevertheless, this is the first approach that occurs to most students who have not truly accepted the Recursion Fairy into their hearts.

Proof by baby-step induction: Let n be an arbitrary non-negative integer. Assume that any non-negative integer less than n can be written as the sum of distinct powers of 2. There are two cases to consider: Either $n = 0$ or $n \geq 1$.

- The base case $n = 0$ is trivial—the elements of the empty set are distinct and sum to zero.
- Suppose $n \geq 1$. The inductive hypothesis implies that $n - 1$ can be written as the sum of distinct powers of 2. Thus, n can be written as the sum of powers of 2, which are distinct except possibly for two copies of 2^0. Let S be this multiset of powers of 2.

  Now consider the following algorithm:

  ```
  i ← 0
  while S has more than one copy of 2^i
      Remove two copies of 2^i from S
      Insert one copy of 2^(i+1) into S
      i ← i + 1
  ```

  Each iteration of this algorithm reduces the cardinality of S by 1, so the algorithm must eventually halt. We claim that for every non-negative integer i, the following invariants are satisfied after the ith iteration of the while loop (or before the algorithm starts if $i = 0$):

  – The elements of S sum to n.

  Proof by induction: Let i be an arbitrary non-negative integer. Assume that for any non-negative integer $j \leq i$, after the jth iteration of the while loop, the elements of S sum to n. If $i = 0$, the elements of S sum to n by definition of S. Otherwise, the induction hypothesis implies that just *before* the ith iteration, the elements of S sum to n; the ith iteration replaces two copies of 2^i with 2^{i+1}, leaving the sum unchanged. □

  – The elements in S are distinct, except possibly for two copies of 2^i.

  Proof by induction: Let i be an arbitrary non-negative integer. Assume that for any non-negative integer $j \leq i$, after the jth iteration of the while loop, the elements of S are distinct except possibly for two copies of 2^j. If $i = 0$, the invariant holds by definition of S. So suppose $i > 0$. The induction hypothesis implies that just *before* the ith iteration, the elements of S are distinct except possibly for two copies of 2^i. If there are two copies of 2^i, the algorithm replaces them both with 2^{i+1}, and the invariant is established; otherwise, the algorithm halts, and the invariant is again established. □

  The second invariant implies that when the algorithm halts, the elements of S are distinct.

In either case, we conclude that n can be written as the sum of distinct powers of 2. □

Repeat after me: "Doctor! Doctor! It hurts when I do this!"

8 Irrational Numbers Exist

Theorem 7. *$\sqrt{2}$ is irrational.*

Proof: I will prove that $p^2 \neq 2q^2$ (and thus $p/q \neq \sqrt{2}$) for all positive integers p and q.

Let p and q be arbitrary positive integers. Assume that for any positive integers $i < p$ and $j < q$, we have $i^2 \neq 2j^2$. Let $i = \lfloor p/2 \rfloor$ and $j = \lfloor q/2 \rfloor$. There are three cases to consider:

- Suppose p is odd. Then $p^2 = (2i+1)^2 = 4i^2 + 4i + 1$ is odd, but $2q^2$ is even.
- Suppose p is even and q is odd. Then $p^2 = 4i^2$ is divisible by 4, but $2q^2 = 2(2j+1)^2 = 4(2j^2+2j)+2$ is not divisible by 4.
- Finally, suppose p and q are both even. The induction hypothesis implies that $i^2 \neq 2j^2$. Thus, $p^2 = 4i^2 \neq 8j^2 = 2q^2$.

In every case, we conclude that $p^2 \neq 2q^2$. □

For some reason, this proof is almost always presented as a proof by infinite descent. Notice that the induction hypothesis assumed that *both* p and q were as small as possible. Notice also that the 'base cases' included every pair of integers p and q where at least one of the integers is odd.

9 Fibonacci Parity

The ***Fibonacci numbers*** $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$ are recursively defined as follows:

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n \geq 2 \end{cases}$$

Theorem 8. *For all non-negative integers n, F_n is even if and only if n is divisible by 3.*

Proof: Let n be an arbitrary non-negative integer. Assume that for all non-negative integers $k < n$, F_k is even if and only if n is divisible by 3. There are three cases to consider: $n = 0$, $n = 1$, and $n \geq 2$.

- If $n = 0$, then n is divisible by 3, and $F_n = 0$ is even.
- If $n = 1$, then n is not divisible by 3, and $F_n = 1$ is odd.
- If $n \geq 2$, there are two subcases to consider: Either n is divisible by 3, or it isn't.
  - Suppose n is divisible by 3. Then neither $n-1$ nor $n-2$ is divisible by 3. Thus, the inductive hypothesis implies that both F_{n-1} and F_{n-2} are odd. So F_n is the sum of two odd numbers, and is therefore even.
  - Suppose n is not divisible by 3. Then exactly one of the numbers $n-1$ and $n-2$ is divisible by 3. Thus, the inductive hypothesis implies that exactly one of the numbers F_{n-1} and F_{n-2} is even, and the other is odd. So F_n is the sum of an even number and an odd number, and is therefore odd.

In all cases, F_n is even if and only if n is divisible by 3. □

10 Recursive Functions

Theorem 9. *Suppose the function $F:\mathbb{N}\to\mathbb{N}$ is defined recursively by setting $F(0)=0$ and $F(n)=1+F(\lfloor n/2\rfloor)$ for every positive integer n. Then for every positive integer n, we have $F(n)=1+\lfloor\log_2 n\rfloor$.*

Proof: Let n be an arbitrary positive integer. Assume that $F(k)=1+\lfloor\log_2 k\rfloor$ for every positive integer $k<n$. There are two cases to consider: Either $n=1$ or $n\geq 2$.

- Suppose $n=1$. Then $F(n)=F(1)=1+F(\lfloor 1/2\rfloor)=1+F(0)=1$ and $1+\lfloor\log_2 n\rfloor = 1+\lfloor\log_2 1\rfloor = 1+\lfloor 0\rfloor = 1$.
- Suppose $n\geq 2$. Because $1\leq\lfloor n/2\rfloor<n$, the induction hypothesis implies that $F(\lfloor n/2\rfloor)=1+\lfloor\log_2\lfloor n/2\rfloor\rfloor$. The definition of $F(n)$ now implies that $F(n)=1+F(\lfloor n/2\rfloor)=2+\lfloor\log_2\lfloor n/2\rfloor\rfloor$.

  Now there are two subcases to consider: n is either even or odd.

  - If n is even, then $\lfloor n/2\rfloor=n/2$, which implies

$$\begin{aligned}F(n)&=2+\lfloor\log_2\lfloor n/2\rfloor\rfloor\\&=2+\lfloor\log_2(n/2)\rfloor\\&=2+\lfloor(\log_2 n)-1\rfloor\\&=2+\lfloor\log_2 n\rfloor-1\\&=1+\lfloor\log_2 n\rfloor.\end{aligned}$$

  - If n is odd, then $\lfloor n/2\rfloor=(n-1)/2$, which implies

$$\begin{aligned}F(n)&=2+\lfloor\log_2\lfloor n/2\rfloor\rfloor\\&=2+\lfloor\log_2((n-1)/2)\rfloor\\&=1+\lfloor\log_2(n-1)\rfloor\\&=1+\lfloor\log_2 n\rfloor\end{aligned}$$

    by the algebra in the even case. Because $n>1$ and n is odd, n cannot be a power of 2; thus, $\lfloor\log_2 n\rfloor=\lfloor\log_2(n-1)\rfloor$.

In all cases, we conclude that $F(n)=1+\lfloor\log_2 n\rfloor$. □

11 Trees

Recall that a *tree* is a connected undirected graph with no cycles. A *subtree* of a tree T is a connected subgraph of T; a *proper* subtree is any tree except T itself.

Theorem 10. *In every tree, the number of vertices is one more than the number of edges.*

This one is actually pretty easy to prove directly from the definition of 'tree': a directed acyclic graph.

Proof: Let T be an arbitrary tree. Choose an arbitrary vertex v of T to be the root, and direct every edge of T outward from v. Because T is connected, every node except v has *at least* one edge directed into it. Because T is acyclic, every node has *at most* one edge directed into it, and no edge is directed into v. Thus, for every node $x\neq v$, there is *exactly* one edge directed into x. We conclude that the number of edges is one less than the number of nodes. □

But we can prove this theorem by induction as well, in several different ways. Each inductive proof is structured around a different recursive definition of 'tree'. First, a tree is either a single node, or two trees joined by an edge.

Proof: Let T be an arbitrary tree. Assume that in any proper subtree of T, the number of vertices is one more than the number of edges. There are two cases to consider: Either T has one vertex, or T has more than one vertex.

- If T has one vertex, then it has no edges.
- Suppose T has more than one vertex. Because T is connected, every pair of vertices is joined by a path. Thus, T must contain at least one edge. Let e be an arbitrary edge of T, and consider the graph $T \setminus e$ obtained by deleting e from T.

  Because T is acyclic, there is no path in $T \setminus e$ between the endpoints of e. Thus, T has at least two connected components. On the other hand, because T is connected, $T \setminus e$ has at most two connected components. Thus, $T \setminus e$ has exactly two connected components; call them A and B.

  Because T is acyclic, subgraphs A and B are also acyclic. Thus, A and B are subtrees of T, and therefore the induction hypothesis implies that $|E(A)| = |V(A)| - 1$ and $|E(B)| = |V(B)| - 1$.

  Because A and B do not share any vertices or edges, we have $|V(T)| = |V(A)| + |V(B)|$ and $|E(T)| = |E(A)| + |E(B)| + 1$.

  Simple algebra now implies that $|E(T)| = |V(T)| - 1$.

In both cases, we conclude that the number of vertices in T is one more than the number of edges in T. □

Second, a tree is a single node connected by edges to a finite set of trees.

Proof: Let T be an arbitrary tree. Assume that in any proper subtree of T, the number of vertices is one more than the number of edges. There are two cases to consider: Either T has one vertex, or T has more than one vertex.

- If T has one vertex, then it has no edges.
- Suppose T has more than one vertex. Let v be an arbitrary vertex of T, and let d be the degree of v. Delete v and all its incident edges from T to obtain a new graph G. This graph has exactly d connected components; call them $G_1, G_2, \ldots, G_d$. Because T is acyclic, every subgraph of T is acyclic. Thus, every subgraph G_i is a proper subtree of G. So the induction hypothesis implies that $|E(G_i)| = |V(G_i)| - 1$ for each i. We conclude that

$$|E(T)| \;=\; d + \sum_{i=1}^{d} |E(G_i)| \;=\; d + \sum_{i=1}^{d} (|V(G_i)| - 1) \;=\; \sum_{i=1}^{d} |V(G_i)| \;=\; |V(T)| - 1.$$

In both cases, we conclude that the number of vertices in T is one more than the number of edges in T. □

But you should ***never*** attempt to argue like this:

Not a Proof: The theorem is clearly true for the 1-node tree. So let T be an arbitrary tree with at least two nodes. Assume inductively that the number of vertices in T is one more than the number of edges in T. Suppose we add one more leaf to T to get a new tree T'. This new tree has one more vertex than T and one more edge than T. Thus, the number of vertices in T' is one more than the number of edges in T'. □

This is not a proof. Every sentence is true, and the connecting logic is correct, but it does not imply the theorem, because it doesn't *explicitly* consider *all possible* trees. Why should the reader believe that their favorite tree can be recursively constructed by adding leaves to a 1-node tree? It's *true*, of course, but that argument doesn't *prove* it.

Here is a *correct* inductive proof using the same underlying idea. In this proof, I don't have to prove that the proof considers arbitrary trees; it says so right there on the first line! As usual, the proof very strongly resembles a recursive algorithm, including a subroutine to find a leaf.

Proof: Let T be an arbitrary tree. Assume that in any proper subtree of T, the number of vertices is one more than the number of edges. There are two cases to consider: Either T has one vertex, or T has more than one vertex.

- If T has one vertex, then it has no edges.
- Otherwise, T must have at least one vertex of degree 1, otherwise known as a leaf.

  Proof: Consider a walk through the graph T that starts at an arbitrary vertex and continues as long as possible without repeating any edge. The walk can never visit the same vertex more than once, because T is acyclic. Whenever the walk visits a vertex of degree at least 2, it can continue further, because that vertex has at least one unvisited edge. But the walk must eventually end, because T is finite. Thus, the walk must eventually reach a vertex of degree 1. □

  Let ℓ be an arbitrary leaf of T, and let T' be the tree obtained by deleting ℓ from T. Then we have the identity

  $$|E(T)| = |E(T')| - 1 = |V(T')| - 2 = |V(T)| - 1,$$

  where the first and third equalities follow from the definition of T', and the second equality follows from the inductive hypothesis.

In both cases, we conclude that the number of vertices in T is one more than the number of edges in T. □

12 Strings

Recall that a *string* is any finite sequence of symbols. More formally, a string is either empty or a single symbol followed by a string. Let $x \cdot y$ denote the concatenation of strings x and y, and let *reverse*(x) denote the reversal of string x. For example, $\mathsf{now} \cdot \mathsf{here} = \mathsf{nowhere}$ and *reverse*$(\mathsf{stop}) = \mathsf{pots}$.

Theorem 11. *For all strings x and y, we have $reverse(x \cdot y) = reverse(y) \cdot reverse(x)$.*

Proof: Let x and y be arbitrary strings. Assume for any proper *substring* w of x that *reverse*$(w \cdot y) =$ *reverse*$(y) \cdot$ *reverse*(w). There are two cases to consider: Either x is the empty string, or not.

- If x is the empty string, then *reverse*(x) is also the empty string, so *reverse*$(x \cdot y) =$ *reverse*$(y) =$ *reverse*$(y) \cdot$ *reverse*(x).
- Suppose x is not the empty string. Then $x = a \cdot w$, for some character a and some string w, and therefore

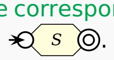
$$\begin{aligned} \mathit{reverse}(x \cdot y) &= \mathit{reverse}((a \cdot w) \cdot y) \\ &= \mathit{reverse}(a \cdot (w \cdot y)) \\ &= \mathit{reverse}(w \cdot y) \cdot a \\ &= \mathit{reverse}(y) \cdot \mathit{reverse}(w) \cdot a && \text{[induction hypothesis]} \\ &= \mathit{reverse}(y) \cdot \mathit{reverse}(x). \end{aligned}$$

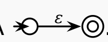
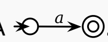
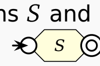
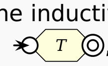
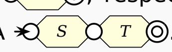
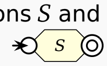
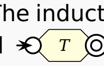
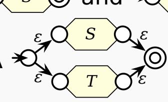
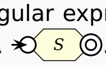
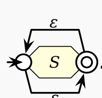
In both cases, we conclude that *reverse*$(x \cdot y) =$ *reverse*$(y) \cdot$ *reverse*(x). □

13 Regular Languages

Theorem 12. *Every regular language is accepted by a non-deterministic finite automaton.*

Proof: In fact, we will show something stronger: Every regular language is accepted by an NFA with exactly one accepting state.

Let R be an arbitrary regular expression over the finite alphabet Σ. Assume that for any sub-expression S of R, the corresponding regular language is accepted by an NFA with one accepting state, denoted . There are six cases to consider—three base cases and three recursive cases—mirroring the recursive definition of a regular expression.

- If $R = \varnothing$, then $L(R) = \varnothing$ is accepted by an NFA with no transitions: .
- If $R = \varepsilon$, then $L(R) = \{\varepsilon\}$ is accepted by the NFA .
- If $R = a$ for some character $a \in \Sigma$, then $L(R) = \{a\}$ is accepted by the NFA .
- Suppose $R = ST$ for some regular expressions S and T. The inductive hypothesis implies that S and T are accepted by NFAs  and , respectively. Then $L(T) = \{uv \mid u \in L(S), v \in L(T)\}$ is accepted by the NFA .
- Suppose $R = S + T$ for some regular expressions S and T. The inductive hypothesis implies that S and T are accepted by NFAs  and , respectively. Then $L(R) = L(S) \cup L(T)$ is accepted by the NFA .
- Finally, suppose $R = S^*$ for some regular expression S. The inductive hypothesis implies that S is accepted by an NFA . Then the language $L(R) = L(S)^*$ is accepted by the NFA .

In every case, $L(x)$ is accepted by an NFA with one accepting state. □