Chapter 17-Proof by Contradiction

Monday, January 9, 2023
9:21 PM

Proof by Contradiction:

Prove that the negative of a statement is false, therefore implying that the statement is true.

Typically used to prove claims that a certain type of object **cannot** exist. The negation of the claim then says that an object of this sort **does** exist.

Simple Example:

Claim 51 *There is no largest even integer.*

Proof: Suppose not. That is, suppose that there were a largest even integer. Let's call it k.

Since k is even, it has the form $2n$, where n is an integer. Consider $k+2$. $k+2 = (2n)+2 = 2(n+1)$. So $k+2$ is even. But $k+2$ is larger than k. This contradicts our assumption that k was the largest even integer. So our original claim must have been true. □

(The phrase "suppose not" is a common indicator used to note the use of proof by contradiction)

More Involved Example:

First, I can (but am too lazy to) prove the claim (*) if k is even, then k^2 is also even.

Now, Claim:

$\sqrt{2}$ is irrational

Proof:

Suppose not. That is, suppose that $\sqrt{2}$ were rational.

Then we can write $\sqrt{2}$ as a fraction $\frac{a}{b}$ where a and b are integers with no common factors.

Since $\sqrt{2} = \frac{a}{b}$, $2 = \frac{a^2}{b^2}$. So $2b^2 = a^2$.

By the definition of even, this means a^2 is even. But then a must be even, by (*) above. So $a = 2n$ for some integer n.

If $a = 2n$ and $2b^2 = a^2$, then $2b^2 = 4n^2$. So $b^2 = 2n^2$. This means that b^2 is even, so b must be even.

We now have a contradiction. a and b were chosen not to have any common factors. But they are both even, i.e. they are both divisible by 2.

Because assuming that $\sqrt{2}$ was rational led to a contradiction, it must be the case that $\sqrt{2}$ is irrational. □

(Note that in proof by contradiction, we often don't prove specific claim true. Instead, we just prove a claim false, or reveal a contradiction that implies a claim false.)

File Compression Example:

In file compression, a ***lossless*** algorithm allows you to reconstruct the *original* file exactly from its compressed version;
But a ***lossy*** algorithm only allows you to reconstruct an *approximation* to the original file.

Claim 52 *A lossless compression algorithm that makes some files smaller must make some (other) files larger.*

Proof: Suppose not. That is, suppose that we had a lossless compression algorithm A that makes some files smaller and does not make any files larger.

Let x be the shortest file whose compressed size is smaller than its original size. (If there are two such files of the same length, pick either at random.) Suppose that the input size of x is m characters.

Suppose that S is the set of distinct files with fewer than m characters. Because x shrinks, A compresses x to a file in S. Because no files smaller than x shrink, each file in S compresses to a file (perhaps the same, perhaps different) in S.

Now we have a problem. A is supposed to be lossless, therefore one-to-one. But A maps a set containing at least $|S| + 1$ files to a set containing $|S|$ files, so the Pigeonhole Principle states that two input files must be mapped to the same output file. This is a contradiction.

(basically saying that if the algorithm is lossless, it has to be one-to-one *(a)*. We feed the smallest file size that **can** shrink *(b)*, called *x*, into the algorithm, then some file with a smaller size, called *y*, has to be the output of *x*. But then if we feed *y* into the algorithm, it cannot be the output itself *(a)*, and cannot shrink *(b)*. Therefore this is a contradiction)
(so the lossless compression method is actually a lie! Sometimes the size gets bigger)