1. Overview

Quick recap: Last time we introduced sets, one of the most fundamental objects in mathematics, and several operations on them. We also introduced functions, which allow us to relate two sets. Most of the properties of functions we discussed only applied to finite sets. We ended last lecture by showing that infinite sets don't behave as intuitively as finite sets. In particular, we showed that $\mathbb{N}$ has a bijection to one of its proper subsets.

In this lecture, we will explain how to think about the cardinalities of infinite sets. Our study of infinite sets will naturally lead to a realization about the existence of computer programs. In particular, we will learn how to prove that there are certain specifications for computer programs that cannot possibly be implemented.

2. Cardinality of Infinite Sets

2.1. **Intro to Infinite Cardinality.** Let's recall some of the definitions and propositions from last time.

Definition 1. Let $f : X \to Y$ be a function.

- f is an injection if for all $x_1, x_2 \in X$, $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$.
- f is a surjection if for all $y \in Y$, there is $x \in X$ so $f(x) = y$.
- f is a bijection if it is an injection and surjection.

Theorem 1. Let X and Y be finite sets. There is a [injection; surjection; bijection] $f : X \to Y$ if and only if $[|X| \leq |Y|; |X| \geq |Y|; |X| = |Y|]$.

At the end of last lecture, we found a bijection $f : \mathbb{N} \to 2\mathbb{N}$ even though $2\mathbb{N}$ is a proper subset of $\mathbb{N}$. A bijection means we have perfectly paired the elements of $\mathbb{N}$ and $2\mathbb{N}$, so in a very real sense these two sets have the same size. The moral of this example is that we can't assign an absolute size to infinite sets in any way that is remotely similar to what we did for finite sets. With finite sets, it was certainly true that a proper subset of a set would have a smaller cardinality than the set itself. With infinite sets, we cannot say this. So how do we define cardinality for infinite sets? The trick is we will define *relative* cardinality using functions.

Definition 2. If X and Y are any two sets, we say $|X| = |Y|$ if there is a bijection $f : X \to Y$.

Let's break down what's happening here. We had a definition of cardinality for finite sets (just the number of elements), but our definition didn't make sense for infinite sets. However, we showed that for finite sets, $|X| = |Y|$ is equivalent to the existence of a bijection $f : X \to Y$, which is a property that makes sense for infinite sets. So, we can **define** $|X| = |Y|$ to mean there is a bijection $f : X \to Y$ for all sets, finite and infinite. This is procedure of extending a definition to a more general case using a property that holds in a simpler case is very common in math. Look out for it!

As an important warning, note that we still have not defined $|X|$ when X is an infinite set. We have only defined what it means for $|X| = |Y|$. In this class we will not give a meaning to $|X|$ for infinite X, so you should never write $|X|$ on its own for infinite X.

Again extending our properties from last time, we can give a meaning to $|X| \leq |Y|$.

Definition 3. If X and Y are any two sets, we say $|X| \leq |Y|$ if there is a injection $f : X \to Y$.

To be clear, $|X| = |Y|$ and $|X| \leq |Y|$ is just notation telling us that we have a bijection or injection. We are **not** saying that some number $|X|$ is equal to or less than some number $|Y|$ since $|X|$ and $|Y|$ are not numbers! Thus, we cannot assume usual properties of $=$ or $\leq$ without proof. However, it turns out almost all of them work as expected (which is why we use this notation!). We can do a few quick examples. First, every bijection is an injection, so $|X| = |Y| \Rightarrow |X| \leq |Y|$. Next, every set has a bijection to itself (the identity map), so $|X| = |X|$. Also, every bijection has an inverse which is a bijection, so $|X| = |Y| \Rightarrow |Y| = |X|$. Here is another that is a bit less obvious.

Proposition 1. Let X, Y, Z be any sets. If $|X| = |Y|$ and $|Y| = |Z|$, then $|X| = |Z|$ (and similarly for $\leq$).

Proof. If $|X| = |Y|$, we know there is a bijection $f : X \to Y$. If $|Y| = |Z|$, we know there is a bijection $g : Y \to Z$. You can check that the composition $g \circ f : X \to Z$ is also a bijection (this is a good exercise!), so $|X| = |Z|$. The same works with $\leq$ and injections. □

2.2. **Countable Sets.** Now that we've established some basic properties of infinite cardinality, let's look at some popular sets to find out which ones have the same cardinality. As a baseline we will use $\mathbb{N}$, which is arguably the simplest infinite set since its elements can be simply listed just by counting.

Definition 4. A set X is countably infinite if $|X| = |\mathbb{N}|$. A set X is countable if X is finite or countably infinite.

So why do we call X countable? Well, by definition, it means we have a bijection $f : \mathbb{N} \to X$. Using f, we can then list all the elements of f just like counting. We just put:

$$f(0), f(1), f(2), \ldots$$

Since f is a bijection, each element of X appears exactly once in the list. Conversely, we may notice that if we can list out the elements of an infinite set, it must be countable. Let's see why with an example.

Consider the integers $\mathbb{Z}$. To show that $\mathbb{Z}$ is countable, let's list its elements. Consider the following list:

$$0, -1, 1, -2, 2, \ldots$$

This list automatically gives us a bijection! Just map the 0th element of the list to 0, the 1st element of the list to 1, the 2nd element of the list to 2, and so on (zero indexing). In other words, we put $f : \mathbb{Z} \to \mathbb{N}$ with $f(0) = 0, f(-1) = 1, f(1) = 2, f(-2) = 3$, and so on. This clearly gives a bijection because every integer maps to a different natural number (its list position), and each natural number is mapped to.

As a warning, we do have to be a little careful when picking our list to show that a set is countable. Consider the following alternate list for $\mathbb{Z}$:

$$0, 1, 2, 3, \ldots, -1, -2, \ldots$$

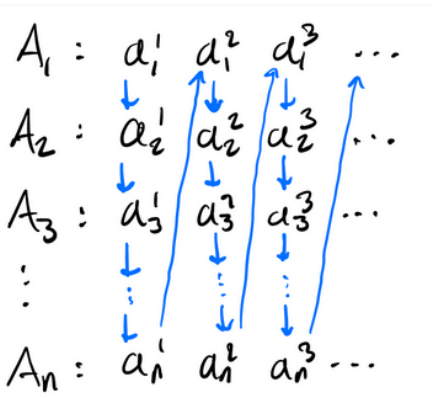
This may look like a good listing of $\mathbb{Z}$, but it has some issues. For instance, what position in the list is -1 at? Actually, since all the nonnegative integers come first, -1 is not at any finite position! So, the list does not represent a bijection $\mathbb{Z} \to \mathbb{N}$. The moral of the story is that providing a listing of a set **is** a valid proof that the set is countable, *but* you must be make sure that every element has a **finite** position in the list. In the first version, every integer (positive or negative) had a finite position, but in the second version, none of the negative integers had a finite position.

Next, let's do another example of a countability proof that's a bit more abstract.

Proposition 2. Let $A_1, A_2, \ldots, A_n$ be disjoint countably infinite sets. Then, their union $\bigcup_{i=1}^{n} A_i$ is countably infinite.

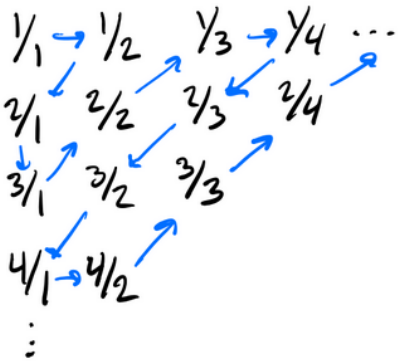
Proof. Since A_i is countably infinite for each i, we can list its elements. Call them $a_i^1, a_i^2, a_i^3, \ldots \in A_i$. Then, we can list all elements of the union as in the picture on the next page.

In other words, we list the first element of each of the n sets, then the second element of each and so on. Since the A_i are disjoint, each element of the union appears exactly once in the list. Each element appears at a finite point in the list since there are finitely many sets A_i. □



Next, let's turn our attention to another important set, the rationals $\mathbb{Q}$. The rationals have properties that suggest there are many more of them than there are integers. For instance, notice that there is no smallest positive rational number. Why not? Well, suppose for contradiction $x \in \mathbb{Q}$ is the smallest positive rational. Then, $\frac{x}{2}$ is a smaller positive rational. The integers certainly don't have any property like this. However, shockingly, the rationals are still countable. Let's attempt to prove it.

For simplicity, let's consider only the positive rationals $\mathbb{Q}^+$. Lay them out in a grid based on their numerator and denominator. Then, we can zig-zag through the rationals as below to create a list.

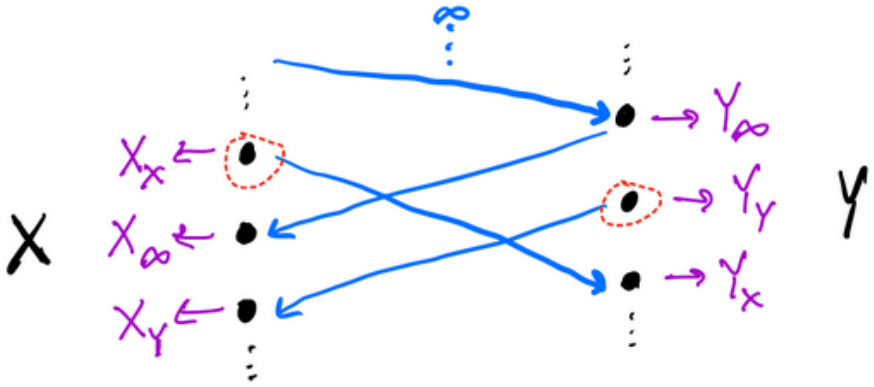


Does this list give a bijection? Frustratingly, it does not! For instance, $\frac{1}{1} = 1$ is the 0th element on our list, but it is also the 3rd since $\frac{2}{2} = 1$. So, should $f(1) = 0$ or 3? Our listing doesn't even give us a well defined function! We can somewhat fix this by defining $f(\frac{p}{q})$ to be the position where $\frac{p}{q}$ *first* appears, so $f(1) = 0$, but now f is not surjective since nothing maps to 3! So, we have only shown $|\mathbb{Q}^+| \leq |\mathbb{N}|$, not $|\mathbb{Q}| = |\mathbb{N}|$. Intuitively, it should be clear that $|\mathbb{Q}^+|$ is not less than $|\mathbb{N}|$, so we should be able to easily conclude that $|\mathbb{Q}^+| = |\mathbb{N}|$. In fact, we can easily get $|\mathbb{N}| \leq |\mathbb{Q}^+|$ since the function $g : \mathbb{N} \to \mathbb{Q}^+$ given by $g(n) = \frac{n+1}{1}$ is an injection.

So, does $|\mathbb{Q}^+| \leq |\mathbb{N}|$ and $|\mathbb{N}| \leq |\mathbb{Q}^+|$ imply $|\mathbb{N}| = |\mathbb{Q}^+|$? If our notation makes sense then it should, but remember that this is really asking if having an injection both ways means there is bijection, since $|\mathbb{Q}^+|$ and $|\mathbb{N}|$ are not actually numbers. Luckily, this statement is true, and is actually an important theorem in set theory. The proof is a bit trickier that most things you'd be asked to come up with yourself, but its not too long so I'll walk through it.

Theorem 2 (Cantor-Schröder-Bernstein)**.** Let X and Y be two sets. If $|X| \leq |Y|$ and $|Y| \leq |X|$, then $|X| = |Y|$.

Proof. Since $|X| \leq |Y|$ there is an injection $f : X \to Y$. Similarly, there is an injection $g : Y \to X$. Since f is an injection, every element $y \in Y$ has **at most** one element $x \in X$ such that $f(x) = y$. Since f is not surjective, there might be no such x. Similarly, every element $x \in X$ has **at most** one element $y \in Y$ so that $g(y) = x$. Using this observation, we can split X into three disjoint sets. Given any $x \in X$, check to see if there is $y \in Y$ which maps to it via g. If not, stop. If there is, go to y. Then, check to see if there is $x' \in X$ that maps to it via f. If not, stop. Otherwise, go to x'. Then, repeat. This process might continue forever, never stopping. If so, we say $x \in X_\infty$. If instead we get stuck on an element of X, we say $x \in X_X$. If we get stuck on an element of Y, we say $x \in X_Y$. Notice that every $x \in X$ is in exactly one of X_∞, X_X, or X_Y. We can define Y_∞, Y_X, and Y_Y in the same way. Below is a picture giving examples of elements in each subset.



Now, we will construct a bijection $h : X \to Y$. We will separately define h on X_∞, X_X, and X_Y. Notice that f maps every element of X_∞ to an element of Y_∞, so we can view f as a function $f : X_\infty \to Y_\infty$. This restricted f is still injective since it is injective on all of X, and the restricted map is surjective since every element $y \in Y_\infty$ is mapped to by f by definition of Y_∞. So, we can define $h(x) = f(x)$ for $x \in X_\infty$. Similarly, the restriction $f : X_X \to Y_X$ is a bijection (try to convince yourself from the picture). So, we can define $h(x) = f(x)$ for $x \in X_X$. Finally, we must define h on X_Y. It is **not** true that $f : X_Y \to Y_Y$ is a bijection, since many element of Y_Y are not mapped to by anything in X. However, $g : Y_Y \to X_Y$ is a bijection since g is injective on all of Y, and every element $x \in X_Y$ must be mapped to by something in Y_Y. But then, we know g has an inverse function $g^{-1} : X_Y \to Y_Y$ which is a bijection. So, put $h(x) = g^{-1}(x)$ for $x \in X_Y$. Combining three parts of our function, we get a bijection $h : X \to Y$. □

Going back to our example with the rationals, we had shown $|\mathbb{Q}^+| \leq |\mathbb{N}|$ and $|\mathbb{N}| \leq |\mathbb{Q}^+|$, so indeed $\mathbb{Q}^+$ is countable. By the same argument, the set of nonpositive (negative or zero) rationals $\mathbb{Q}^{\leq 0}$ is countable. But then, $\mathbb{Q} = \mathbb{Q}^+ \cup \mathbb{Q}^{\leq 0}$ is countable since it is the union of disjoint countable sets.

Overall, the Cantor-Schröder-Bernstein theorem is a very useful tool in countability proofs. It is often very easy to find an injection $f : \mathbb{N} \to X$ for your infinite set X, and by CSB you need only find an injection $X \to \mathbb{N}$ instead of a bijection.

2.3. **Uncountable Sets.** To finish our discussion of cardinality, let's look at one final example: the real numbers $\mathbb{R}$. So far, all of our important sets of number have been countable. Before we evaluate $\mathbb{R}$, let's be clear about what a real number is. For our purposes, we will think of a real

number as any number with an infinite decimal expansion. For instance,

$$2 = 2.0000000000\cdots$$
$$\frac{1}{3} = 0.3333333333\cdots$$
$$\pi = 3.1415926535\cdots$$

are all real numbers. We can tell 2 and $\frac{1}{3}$ are rational from their decimal expansions because after some finite point the digits make a repeating pattern (in this case, we just get a single repeating digit). π is not rational because its decimal expansion never repeats. We will not prove the connection between repeating decimals and rationality, or that π is irrational; this is purely to give an intuitive sense of what extra elements we have added going from $\mathbb{Q}$ to $\mathbb{R}$. The addition of non-repeating decimals may not seem like much, but we will see this is finally the breaking point where our set is uncountable.

To show $\mathbb{R}$ is uncountable, we need to show there is no bijection $f : \mathbb{N} \to \mathbb{R}$. To do this, we will use **Cantor's diagonalization argument**. The idea is that any function $f : \mathbb{N} \to \mathbb{R}$ gives us a list of real numbers which should be comprehensive if f is a surjection. Then, we will use the list to construct a real number not in the list, which shows f is not surjective, and hence not a bijection. In fact, we will show there is no surjection $f : \mathbb{N} \to (0, 1)$, where $(0, 1) = \{x \in \mathbb{R} \mid 0 < x < 1\}$. This shows that $(0, 1)$ is uncountable and implies that $\mathbb{R}$ is uncountable, since if there is no surjection $\mathbb{N} \to (0, 1)$ there cannot possibly be one $\mathbb{N} \to \mathbb{R}$ (and hence there is no bijection $\mathbb{N} \to \mathbb{R}$ either).

Theorem 3. $\mathbb{R}$ is uncountable.

Proof. Following the outline above, consider any function $f : \mathbb{N} \to (0, 1)$. We will show f cannot be surjective. To see this, consider making a vertical list of the elements in the range of f in order, and circle the digit in the nth decimal place of the nth item. For instance, consider the example below.

$$f(0) = 0.\boxed{1}8942\cdots$$
$$f(1) = 0.3\boxed{0}664\cdots$$
$$f(2) = 0.01\boxed{8}97\cdots$$
$$\vdots$$

Then, make a new real number in $(0, 1)$ by taking each circled digit and changing the nonzero digits to 0, and changing the zeros to 1. For instance, from the above example, we get $x = 0.010\cdots$. Notice $x \neq f(0)$ since it is different in the 1st decimal place. $x \neq f(1)$ because it is different in the second decimal place, and so on. So, $x \neq f(n)$ for any $n \in \mathbb{N}$, meaning f is not surjective. So, there is no surjection $\mathbb{N} \to (0, 1)$, hence no surjection $\mathbb{N} \to \mathbb{R}$, hence no bijection $\mathbb{N} \to \mathbb{R}$. So, $\mathbb{R}$ is uncountable. □

2.4. **Summary and Final Example.** Let's wrap up our discussion of infinite sets. Our main goal for infinite sets is to decide whether they are countable or uncountable. To show a set it countable, we come up with a list of its elements, or we find a bijection to another countable set. We can also make our lives easier by using the CSB theorem and finding injections both ways instead.

To show a set is uncountable, we either find a bijection to another uncountable set like $\mathbb{R}$ (or find injections both ways with CSB), or we perform a diagonalization argument: write out a list of the set's elements, and construct an element missing from the list.

Is there a way to tell if a set is countable before we start a proof? Kind of. There are two rough criteria that a set must pass to be uncountable. First, there should be infinitely many elements. Second, the elements of the set should require an infinite amount of information to describe them.

For instance, to describe a real number, we had to specify an infinite decimal expansion, whereas rationals could be specified with just two integers.

As one final example, consider the set of functions $\mathbb{N} \to \mathbb{N}$, denoted $F(\mathbb{N}, \mathbb{N})$. There are infinitely many functions, and each one requires infinite information to specify, since we must pick $f(0), f(1), f(2), \ldots$ and so on. So, we might guess the set of functions is uncountable. In fact, it is! We can prove it by a very similar diagonalization argument. Just make any list of functions $f_0, f_1, f_2, \ldots \in F(\mathbb{N}, \mathbb{N})$ and then make a new function f so $f(0) \neq f_0(0)$, $f(1) \neq f_1(1)$, and so on. f differs from every function in the list, so it is missing from the list. Thus, every list is missing an element, so there is no bijection $\mathbb{N} \to F(\mathbb{N}, \mathbb{N})$. We draw a picture of the diagonalization below. As before, each horizontal line is one element, and we used the boxed pieces to construct a new different element.

$$\begin{array}{lllll} f_0: & \boxed{f_0(0)} & f_0(1) & f_0(2) & \cdots \\ f_1: & f_1(0) & \boxed{f_1(1)} & f_1(2) & \cdots \\ f_2: & f_2(0) & f_2(1) & \boxed{f_2(2)} & \cdots \\ \vdots & & & & \end{array}$$

It can be shown that there is a bijection between $F(\mathbb{N}, \mathbb{N})$ and $\mathbb{R}$, so they have the same size. It is also not too hard to find sets that are bigger than $\mathbb{R}$. But are there any sets smaller than $\mathbb{R}$ that are still uncountable? This question is called the **continuum hypothesis**, and we will briefly discuss it at the end of the lecture.

In the meantime, the uncountability of functions $\mathbb{N} \to \mathbb{N}$ will actually serve as crucial motivation for our last topic of the lecture.

3. Computability

3.1. **Motivation.** To understand the significance of the uncountability of the functions $\mathbb{N} \to \mathbb{N}$, we must spend a bit of time thinking about computer programs. In this class, we will think of a **computer program** as a piece of text (in some language) that implements a function. The program may have side effects (like printing something), but we will say it successfully implements a function f if, when given input x, the program returns $f(x)$. How many computer programs are there? Every computer program can be converted to a binary string (in fact, this is what your computer does when it interprets/compiles code), so we can think of a computer program as a binary string. It must be a finite length binary string, since it is physically stored on your computer. The set of finite length binary strings is countable: just list them by length. So, there are finitely many computer programs.

On the other hand, how many functions could we want to implement? We can think of a program's input and output as a binary string too, so there are a countably infinite set of possible inputs and outputs. Thus, the number of functions $\{\text{possible inputs}\} \to \{\text{possible outputs}\}$ is the same as the number of functions $\mathbb{N} \to \mathbb{N}$, which we showed is uncountable! So, not all possible functions from inputs to outputs can be implemented as computer programs. We call these functions **uncomputable**.

3.2. **The Halting Problem.** The next natural thing to do is try to come up with an example of a function that cannot be implemented. Unsurprisingly, we have to get somewhat creative to come up with an uncomputable function; most typical problems are solvable with a computer program, even if the program is very slow.

We will take some inspiration from the mathematical concept of **self-reference**. Consider the set of sets $R = \{S \text{ a set} \mid S \notin S\}$. So, R contains all sets which do not contain themselves. Is $R \in R$? If so, then by definition of R, $R \notin R$, a contradiction. If $R \notin R$, then by definition of R,

$R \in R$, another contradiction! So, the only reasonable conclusion is that R does not exist. This example is called **Russel's paradox**, and is the the reason we must be careful about building infinite sets that contain other sets.

We will use this example as inspiration to build our uncomputable function. To create a self-referential paradox, we will consider a function which takes computer programs and inputs as input. Consider the following function, TestHalt, which takes in the binary representation of a program P, and the binary representation of an input x to program P.

$$\text{TestHalt}(P, x) = \begin{cases} 1, & P(x) \text{ halts in finite time} \\ 0, & P(x) \text{ never halts} \end{cases}$$

We claim this function is uncomputable. Before we prove it, why might we think this function will be uncomputable? Well, suppose someone gives you a complicated piece of code and an input and runs it. After it runs for a while, you may wonder if the code is stuck in an infinite loop, or if it is just slow. Although you might be able to spend a while staring at the code and figure out which is happening, it seems difficult to come up with a general algorithm for checking. But how will we actually prove there is no program that can implement TestHalf? In the spirit of self-reference and Russell's paradox, we will suppose we have code that implements TestHalt, and then come up with another computer program and input that gives us a contradiction both if it halts and if it does not, a contradiction. Let's do it.

Theorem 4 (Turing)**.** TestHalt is uncomputable.

Proof. For the sake of contradiction, suppose we have written a computer program which implements TestHalt. We will use TestHalt as a helper function to write another computer program called Turing as follows. As usual, Turing takes a binary string as input. But here, we interpret that binary string as both another program itself, and as an input string to that program.

```
def Turing(P):
    if TestHalt(P,P) == 1: # P(P) halts in finite time
        enter an infinite loop
    else: # P(P) loops forever
        return
```

We are now ready for our self-reference contradiction. Consider passing the binary string representation of Turing into Turing. If Turing(Turing) halts in finite time, then by definition TestHalt(Turing, Turing)= 1. But then, looking at the code of Turing, we see Turing(Turing) enters an infinite loop. Contradiction! Now instead suppose Turing(Turing) loops forever. By definition, TestHalt(Turing,Turing)= 0, so looking at the code of Turing, Turing(Turing) halts in finite time. A contradiction again! So, the program Turing cannot exist. But we made Turing with only basic programming language operations and TestHalt. So, the program that implements TestHalt cannot exist! □

We've found an uncomputable function! But are there others? It turns out that there are many uncomputable functions, including many that seem much easier to implement than the Halting Problem. For instance, consider the function HaltsOnZero which takes in the binary encoding of a program P and is defined as follows.

$$\text{HaltsOnZero}(P) = \begin{cases} 1, & P(0) \text{ halts in finite time} \\ 0, & P(0) \text{ never halts} \end{cases}$$

This seems much easier to implement than TestHalt since we only need to check the input 0, not an arbitrary input. However, we can show HaltsOnZero is also uncomputable. Do we need to do

a confusing self-reference proof again? No! Instead, our strategy will be a proof by contradiction where we assume code for HaltsOnZero exists, and then we use it and basic programming features to implement TestHalt. Since we know TestHalt cannot be implemented, this is a contradiction! Thus we can conclude HaltsOnZero does not exist. This type of proof is called a **reduction** from TestHalt to HaltsOnZero, because we are reducing the problem of implementing TestHalt to the problem of implementing HaltsOnZero. Not that it is not a reduction of HaltsOnZero to TestHalt, since we are using HaltsOnZero to implement TestHalt, not the other way around. Let's do it!

Proposition 3. HaltsOnZero is uncomputable.

Proof. For the sake of contradiction, suppose we have implemented HaltsOnZero. We want to implement TestHalt. The key is HaltsOnZero checks if a program halts on input 0. So, to implement TestHalt(P, x), we will write another program called Helper(n) so that Helper(0) halts if and only if $P(x)$ halts. Thus, we can use HaltsOnZero to implement TestHalt.

```
def TestHalt(P,x):
    def Helper(n): # Helper(0) halts if and only if P(x) halts
        if n == 0:
            run P(x)
        return
    return HaltsOnZero(Helper)
```

Let's break this down. Since Helper(0) runs $P(x)$ and then returns, Helper will halt on input 0 if and only if $P(x)$ halts. So, if $P(x)$ halts, HaltsOnZero(Helper) will return 1, and so TestHalt will return 1. If $P(x)$ never halts, HaltsOnZero(Helper) will return 0, and so TestHalt will return 0. Thus, the above code implements TestHalt correctly! This is a contradiction, so HaltsOnZero must not exist. As a note, we didn't even need the if statement in Helper since we don't care what Helper does on inputs besides 0; the code for Helper could just run $P(x)$ and then return. □

So, there are other uncomputable functions that we can think of! In general, how can we tell just by looking if a function is uncomputable? There isn't any exact criteria, but most that we will deal with in this class take programs as input, and check if the input program halts or produces a particular output (or something similar). The idea is this is impossible to implement because we cannot tell if a program is looping forever or is about to finish.

3.3. **Summary and Incompleteness.** In this lecture, we introduced a way of comparing the cardinality of infinite sets using functions. We defined countable sets, which are sets with the same size as $\mathbb{N}$ and can have their elements listed. We also saw uncountable sets such as $\mathbb{R}$, and learned how to prove uncountability using the diagonalization argument. We also learned the CSB theorem which gives us an easier way to prove two sets are the same size. Finally, we talked about computer programs and uncomputable functions, and showed that TestHalt is uncomputable. We learned how to use reductions from TestHalt to show other functions are uncomputable.

To finish this lecture, let's return to the continuum hypothesis: is there an uncountable set which is smaller than $\mathbb{R}$? Amazingly, this question turns out to be unanswerable within the commonly accepted modern mathematical framework. Specifically, modern mathematics is based on a list of **axioms** about sets called the ZFC axioms, which are statements we assume to be true. It can be shown that neither answer to the continuum hypothesis contradicts the ZFC axioms; either answer is acceptable, and there isn't a great reason to prefer one way over the other. When a system of mathematics has a theorem that cannot be proved or disproved, it is called **incomplete**. Incompleteness may seem very bad, but Gödel showed that it is actually inevitable. Gödel's incompleteness theorem is the statement that any mathematical system which is complex enough to describe arithmetic is

either incomplete, or inconsistent (meaning we can start with the axioms and derive a contradiction). Inconsistency is definitely something to avoid, so incompleteness doesn't seem so bad! The proof of Gödel's theorem is somewhat involved, but it is similar in spirit to Russel's paradox and the uncomputability of TestHalt.