And now for some impossibility proofs

So what's the goal of the week? [insert video bicycle kick] The idea is to show that for certain problems ***there is no computer program that can solve them***. I have to confess in advance, that at the end of the day, this doesn't really have much practical impact. But it is at least a noble endeavor, and at least puts some limits on the power of computation. And it's good for you. Or at least I get paid for it. Hopefully.

One significant point in our history was 1900. In honor of the the turn of the century, a mathematician named Hilbert proferred a list of the most important unsolved problems of the time. One of the problems he listed was to give (what we call) an ***algorithm*** for solving Diophantine problems. A ***Diophantine problem*** is where one has an equation and is only interested in integer solutions (integer meaning whole number). For example, solutions to $x^2 + y^2 = z^2$ are called Pythagorean triples. [I wonder why?]

As a marginal aside, the question of solutions to $x^m + y^m = z^m$ for m bigger than 2 was long the focus of work. Fermat's Last Theorem states that there is no solution with positive integers. Though it has that name because Fermat claimed (around 1637) that he knew a short proof, the (first) proof was given only about 25 years ago, and took hundreds of pages and used ideas from many areas of mathematics (and got Andrew Wiles a knighthood).

Anyway, it took a while but people started asking: maybe there is no algorithm for Diophantine equations. Meaning: a procedure that would take any equation as input and was guaranteed to halt and was guaranteed to give the right answer as to whether there was an integer solution or not. There is a trivial idea that is guaranteed to halt if the answer is yes: try all possibilities in some systematic fashion (a la breadth-first-search). And it's trivial to solve some equations. But the algorithm has to be perfect.

Skipping a few years, we move to the Institute for Advanced Studies at Princeton. In the 1930s you have people like Einstein and von Neumann hanging around, so it was quite intellectually stimulating. People like Alonzo Church had begun asking: how does one in general prove that an algorithm for a problem is not possible? One idea was to provide a model of computation, and then

prove that in that model certain things are impossible. In his case Church offered lambda calculus; he had Alan Turing as a Ph.D. student who provided our beloved Turing Machine.

But slightly earlier, we have Kurt Gödel. He was looking at the power of axioms for mathematical systems. One of his famous Incompleteness Theorems, (essentially) is that for any sufficiently powerful collection of axioms for mathematics, there will always be ***true statements that do not have a proof***. The tool he used was to provide a way to encode a statement about numbers as a single number, and then to use (something akin to) self-reference to produce a contradiction. It was one of those watershed moments in mathematics. Earlier, people had spent painstaking time to try to come up with the definitive set of axioms, and try to get mathematics absolutely properly founded. This included work by Russell (of not-barber fame) and Whitehead who published a book in 1910 that took over 350 pages to reach the proof that $1 + 1 = 2$.

The idea of self-reference first showed up in a 1879 paper by Georg Cantor. Cantor showed that there are different sizes of infinity. The proof used a method of self-reference which is called ***diagonalization*** (and we will see why). Essentially, diagonalization is the ONLY tool we humans know of to provide impossibility proofs in theoretical computer science. Also, Cantor's theorem has an important direct implication for us, as we shall see.

So, as an analogous result to Gödel's incompleteness theorem, we have the result that given any reasonable model of computation, there will always be questions about that model that that model cannot solve.

Oh, and finally in 1970 it was showed by Matiyasevich that there is no algorithm for solving Diophantine equations. [Insert plug for Julia Robinson.]