Why Maths is incomplete (and maybe is meant to be)

Ashrith Agastyaraju

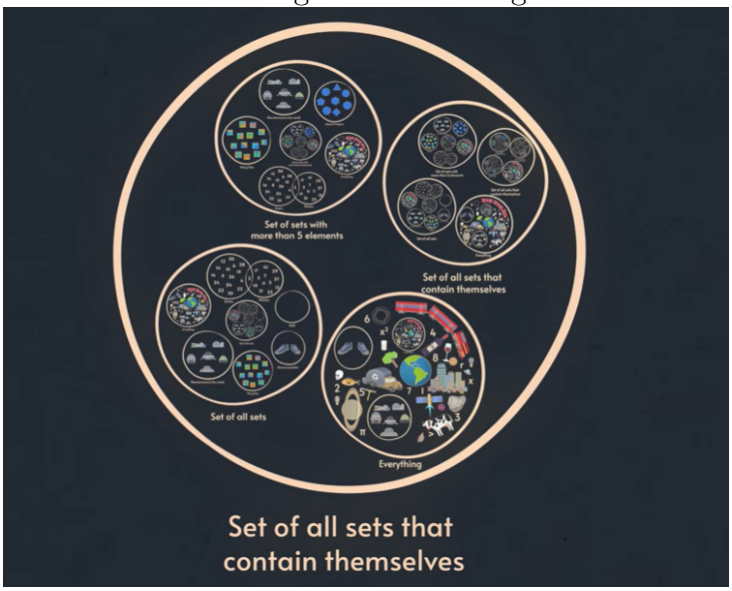
March 31, 2023

1 Introduction

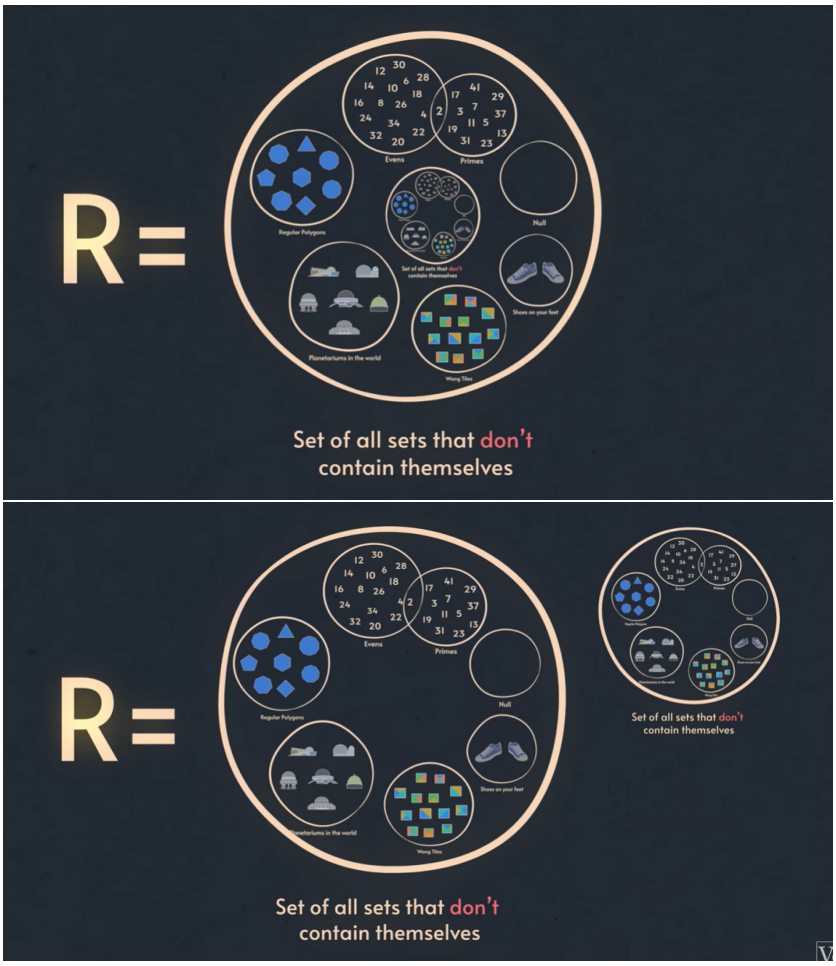
Imagine that every public library had to make a catalogue of all its books (a book that contained a list of all the books in the library). As the catalogue is part of the books in the library, a librarian decides to include it in the catalogue for completeness' sake; while another librarian decides to leave it out as the catalogue, being one of the library's books, is obvious.

Then, let us say that we have the master librarian who makes two master catalogues : one of all the catalogues that list themselves and one of all those that don't. However, the issue for the master librarian is whether these master catalogues should list themselves. For "our catalogue of all catalogues that list themselves", there is no problem. Whether or not the librarian includes it, it remains a true catalogue of "those that list themselves". But the master librarian will fail with the second catalogue: "the catalogue of all catalogues that don't list themselves". The librarian cannot include it in its own listing otherwise it would belong to the other catalogue. If the master librarian does not include it, it would become an incomplete catalogue and the master librarian would be sacked!

E.g. This would be our catalogue of all catalogues that list themselves



But the problem arises in choosing where "the catalogue of all catalogues that don't list themselves" or the set of all sets the do not contain themselves goes. If set R contains itself, it must not contain itself. If R does not contain itself, then it must contain itself.[1] This is a paradox!



In the dawn of the early 20th century, mathematics began to rely more on set theory (which is like the catalogues we sorted), the contradictions it implied became more apparent. Above is a paradox that the mathematician Bertrand Russell published to highlight the need for there to be more precise rules for what kinds of reasoning were to be allowed and what were not. At a time when mathematics was perceived as being objective truth, some were frightened by these paradoxes.

One of these was David Hilbert, a formalist who believed that mathematics could be treated as if it were a game. Using this analogy, a game of chess would be the same as "doing maths", but statements about chess correspond to statements about the scope of mathematics (i.e. what can maths do ?). It was statements like these that Hilbert focused on in his programme, with the aim of answering three main questions:

1) Is mathematics complete (i.e. can you prove every true statement)? If we were given a set of rules, can we prove or disprove every mathematical statement?
2) Is mathematics consistent(with no contradictions)? Or is it riddled with paradoxes like that which Russell posed?
3) Is every statement in mathematics decidable? Is there a "miracle machine" that when given a mathematical statement would tell us whether it was provable or not? This was famously called the Entsheidungsproblem (in German: the decision problem)[2].

I will start off by introducing the beautiful framework made by Alan Turing for solving the 3^{rd} problem, then we will use this to answer the other two questions that Hilbert's programme was set up to answer.

2 Turing machines

> *"No, I'm not interested in developing a powerful brain."*
> *Alan Turing*

Before we move onto how Turing solved the 3^{rd} main question of Hilbert's programme, we must understand what Turing created to solve this problem. Turing set out to find a solution in the simplest way possible, by making a machine that was capable of following a set of instructions and show an output based on a given input. However, he did not concern himself with the *physical* details, rather started by focusing on the rules that would govern its *computation*. One could call this the beginning of abstraction as a method of computational thinking; simplifying the problem by generalizing it and removing unnecessary details.

The first thing that came to mind for Turing was the typewriter. Turing had wondered what made it mechanical in the first place.

Of course, it was the response depending on the current state of the machine (e.g. the lowercase and uppercase). He abstracted this to consider machines with a finite number of possible states, each an exactly determined action carried out. To simplify it further, he imagined these typewriters only working on one infinite line, or tape, marked off into squares. It would have the ability to read from and write into a square. But here is what made Turing machines automatic: based on the state of the machine and the symbol scanned, there would be different behaviour from the machine (with each action being written in a behaviour table).

Figure 1: Scanner pointing towards 0 [3]

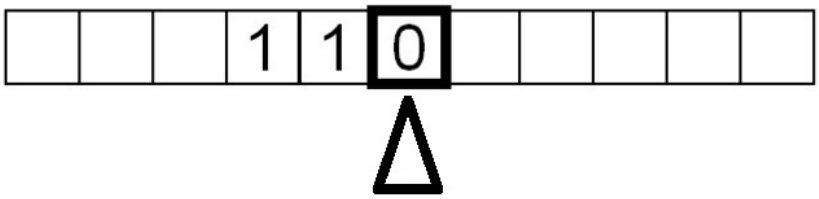
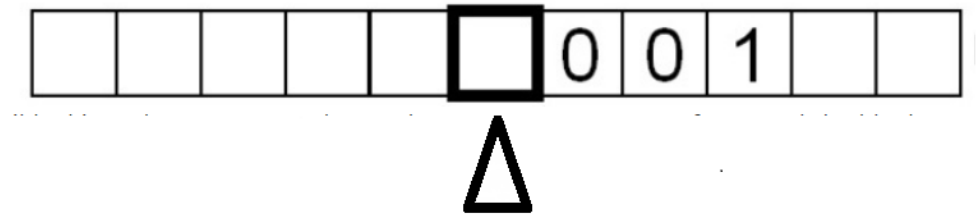


Figure 2: our behaviour table

Symbol read	**Write instruction**	**Read instruction**
Blank	None	None
0	Write 1	Move scanner to the left
1	Write 0	Move scanner to the left

For example, we have a Turing machine in Figure 1, pointing at a 0. We can apply these actions in our behaviour table depending on the symbol scanned (so our first action would be to write 1 onto that square and move the scanner to the left). Applying the instructions in our behaviour table until there is no instruction to do gives us Figure 3.

Figure 3: Scanner pointing towards a blank number



3 The Halting problem and what it shows

Could there be a machine that answers Hilbert's decision problem?
That was the question that gripped Turing's mind as he ran through the never-ending Grantchester meadow. Turing had realised that any 'real number' (all integers, irrationals and rationals) defined by some rule could be calculated by one of his machines (given infinite time). Turing machines were a theoretical model for the "computers" (people who performed calculations) in the 1900s. The property of being able to do anything that a Turing machine or a "computer" could do was called Turing completeness. Any Turing machine, or Turing complete entity had to follow manipulations according to a table of rules like in Figure 2. This brought up a crucial question: what if this set of instructions gave an infinite number? For instance: π being an infinite decimal means that the machine would never stop running, requiring an infinite amount of 'tape'.

However, the Turing machine would find each of the *digits* in finite time, only using a finite amount of tape. This meant that Turing had a way of representing infinite decimals like π and e using finite behaviour tables like in Figure 2. These numbers for which there was a Turing machine that could find out the n^{th} digit of that number were known as **computable** numbers. Despite the effectiveness of Turing machines, Turing was fascinated with the notion of an **uncomputable** number, where the Turing machine would keep running. In that case, Turing asked: would the machine ever stop, and if it did how could we predict the machine stopping? Could it be decided in advance? This is the Halting problem: the question that asks whether everything as we know it is decidable. But before we see how Turing solved the Halting problem, we need to look at how a brilliant mathematician proved that some infinities are bigger than others- through Cantor's Diagonal.

3.1 Cantor's Diagonal argument

"A set is a Many that allows itself to be thought of as a One."
- Georg Cantor

Fifty years before Turing machines were envisioned, the mathematician Georg Cantor had thought of something called Cantor's diagonal argument, which was a proof that not only proved the existence of irrational numbers (numbers that cannot be expressed as a fraction), but also proved that the number of real numbers (irrationals and rationals) was actually greater than the number of rational numbers (numbers that can be expressed as a fraction).

We will start with an infinite list of **rational** numbers:

1/2	**0**.	5	0	0	0	0	...
1/3	0.	**3**	3	3	3	3	...
1/4	0.	2	**5**	0	0	0	...
1/5	0.	2	0	**0**	0	0	...
1/6	0.	1	6	6	**6**	6	...
1/7	0.	1	4	2	8	**5**	...
1/8	0.	1	2	5	0	0	**0**	...

and so on...

If we took the decimal created from the diagonal digits, we get a decimal $n = 0.350650....$ Now if I added one to each of the digits, we get a new decimal $m = 1.461761....$ What Cantor discovered was that if we keep going through our infinite list of rational numbers, we would find that this new decimal m would not be in our infinite list of rationals.

To see why, let's suppose that m was part of our list of rational numbers. Since m is a rational number, we know that it has a decimal expansion that either terminates or repeats. However, we can also see that m cannot have a terminating or repeating decimal expansion. This is because adding 1 to each digit of n ensures that m differs from every number in the list in at least one decimal place. Therefore, if we write out the decimal expansion of m, we will get an infinite sequence of non-repeating digits- an irrational. Using this Cantor proved that the infinite set of reals cannot be in one-to-one correspondence with the infinite set of rationals, rather one could "discover" new irrational numbers from rationals. The concept of the irrational rising from the rational will become important later on.

3.2 Proving the unsolvability of the Halting problem

Assuming that there already was a Turing machine that could solve Hilbert's question, Turing set to work. We will start similarly to Cantor's Diagonal argument, given that we have an infinite set of Turing machines, let us say that there is a Turing machine $T(n;m)$ that shows us the output of the n^{th} Turing machine when given a number m to work with. Each row shows us the output of the n^{th} Turing machine as it is applied to each of the natural numbers $0, 1, 2, 3, 4, 5, 6....$
Therefore, each row is a **computable** sequence as each *digit* in each row has been generated from a Turing machine. To account for Turing machines that keep going (like the one that generates π), we have multiplied every digit in our table by a new number, x where

$$x = \begin{Bmatrix} 0 : \text{If the } n^{th} \text{ Turing machine does not stop} \\ 1 : \text{If the } n^{th} \text{ Turing machine does stop} \end{Bmatrix}$$

This ensures that every digit in our table has a value (if our Turing machine never stopped, no value would be outputted). Similar to Turing's actual proof, we will say that each output comes from a function:

$$S_n(m) = x * T_n(m)$$

$m->$ $n\downarrow$	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1
3	0	1	0	1	0	1	0	1	0
4	0	1	0	1	0	1	0	2	0
5	0	0	0	0	0	0	0	0	0
6	0	1	2	3	4	5	6	7	8
7	0	1	1	2	3	5	8	13	21
8	0	1	0	0	1	0	0	0	1

Figure 4: Our table of outputs where n^{th} Turing machine is on the left column, and the natural number m is on the top.

By definition, every **computable** sequence of natural numbers must exist in each row, as each digit in each row has been generated using a Turing machine following some set of instructions – an algorithm.

Here is where Turing's genius came into play: He started by considering the diagonal digits in our table: [0 ,0 ,1 ,1 ,0 ,0 ,6 ,3 ,1 ...] from Figure 5 making our new function:

0	0	0	0	0	0	0	0	0
0	**0**	0	0	0	0	0	0	0
1	1	**1**	1	1	1	1	1	1
0	1	0	**1**	0	1	0	1	0
0	1	0	1	**0**	1	0	2	0
0	0	0	0	0	**0**	0	0	0
0	1	2	3	4	5	**6**	7	8
0	1	1	2	3	5	8	**3**	2
0	1	0	0	1	0	0	0	**1**

Figure 5: All our diagonals

To each of those digits he added one, giving us a new sequence:
$[1, 1, 2, 2, 1, 1, 7, 4, 2, ...]$.
This also has to be a computable sequence as a Turing machine is capable of following an algorithm to do this. But, this sequence s cannot be part of our original table because it differs from each element by 1. If it were part of our original table then using our formula from Figure 4:

$$x * T_n(m) + 1 = x * T_n(m)$$
$$=> 1 = 0$$

which clearly cannot be true!

Just as Cantor's diagonal showed how the rational gave rise to the irrational, Turing used the diagonal argument to show how the *computable* gave rise to the *uncomputable*. Therefore, Turing realised that there was **no "definite way" of solving all mathematical questions**, since an uncomputable number is an unsolvable problem.[1]
To summarise, Turing first assumed that there was a Turing machine that decided whether or not the n^{th} Turing machine would stop, and from that assumption came across a contradiction — hence a proof by contradiction. This proved that there could not be a machine that could decide the provability of any mathematical statement, delivering a devastating blow to Hilbert and the formalists.

[1]To be historically accurate, the American mathematician Alonzo Church also independently proved the unsolvability of the Halting problem (at the same time as Turing) using his own framework of simple functions called lambda calculus.

4 A quick glance at Gödel

We have looked at the Halting problem and Turing's simple but elegant framework he used to prove its unsolvability. Now we will use the idea of Turing machines to answer the main question of this essay — answering whether maths is incomplete and inconsistent.

To find whether mathematics was complete (where every true statement can be proved) and consistent (where there are no contradictions), Gödel needed to assign a number to each variable, hence allowing for conversion between numbers and formulae. But Gödel did not just want them to be variables, he wanted it so that every possible formula could be represented by a single number. What numbers can uniquely be factored out? Prime numbers of course, since any number can be uniquely factored into primes. With these ideas, Gödel could represent every possible mathematical statement with a single number. Have we seen something like this before? We have with Turing machines! Every Turing machine manipulates symbols on a tape according to a set of rules, meaning there is a unique Gödel number for every possible Turing machine (in Turing's original proof of the Halting problem he relied on Gödel numbering to show the contradiction).

We will use the powerful proof by contradiction again to find out whether there could be a complete system of mathematics. So let us start by saying we did have a complete and consistent system F, which I can say is powerful enough to deal with Turing machines. Then I claim that this system F can easily solve the Halting problem. Say we are already given a Turing machine M and we want to find out whether it halts on a blank piece of tape. Because F is complete, there would be a proof of whether this Turing machine M halts or not. As this system F is perfectly consistent - where every provable statement is true - the proof that it gives us must be true. However, since Alan Turing already proved that the Halting problem is undecidable, such a system F could not have existed! This gives us our first incompleteness theorem: Every system of mathematics has statements that can neither be proved nor disproved in that system [4]. An example of this would be the statement "Turing machine M halts on blank tape"- which is neither provable or disprovable. Using similar logic, we cannot prove that system F is consistent by using the rules in system F itself.

5 Meant to be incomplete

"Either mathematics is too big for the human mind or the human mind is more than a machine."
Kurt Gödel

So we have established that for any set of axioms in a system, there will be statements that are neither provable nor disprovable. Gödel's result had shook the foundations of mathematics, showing us that systems like simple arithmetic were also incomplete; dooming Hilbert's programme once and for all. Although Gödel's work built the foundations for Turing and Church to follow, I feel that the ideas behind Turing machines encompass much more than just a response to Gödel. To me, they show the quintessential nature of what it means to be human- to think rationally. The halting problem reminds me that there are limits to what this rationality and deductive thinking can do.

Undecidability is not limited to simple Turing machines and programs, rather it can crop up anywhere where the system can recognise any sort of rules. Turing completeness, or the ability to follow a set of instructions, is a property of such systems. An interesting example would be musical notation, which is Turing complete through a series of transpositions and repeated notes with our tape becoming an infinite music staff[2]. However, every Turing complete system has its own rendition of the Halting problem. In this case, it is whether the music will stop playing.

When I talked about "computers" (people who performed calculations at breakneck speed), I did not mention that the Turing machines we use daily were designed to capture the mental abilities of these "computers". Therefore, we ourselves are Turing complete, for the concept itself was envisioned to replicate the powers of the human mind! So, are the lives we lead Turing complete? They must be, as we enter this world with our own rendition of the Halting problem like all things Turing complete: we cannot decide when we enter and leave this world.

[2]See the Choon programming language:
https://gwern.net/doc/www/esoteric.sange.fi/ceba1ccc6cb04065c7167d21071c5ee74fc16e50.html

Perhaps this is the importance of undecidability, no method of reasoning is complete nor can it be shown to be consistent using the same reasoning. Us humans can try to get closer to completion, but we will never reach it. Maybe, such incompleteness is meant to be, for without it what meaning would our lives have? On second thought, undecidability does not seem so bad, as it helps us to know the limits to which we can answer the unanswerable. Even with this incompleteness and undecidability, we will keep trying to understand the world better and build even greater machines. This pursuit of completeness might just be what gives our life meaning.

6 References

[1] Pictures explaining Russell's Paradox www.youtube.com. (n.d.). Math's Fundamental Flaw. [online] Available at: https://www.youtube.com/watch=HeQX2HjkcNot=1734s [Accessed 22 Mar. 2023].

[2] Fand (2021). From Hilbert With Love — Entscheidungsproblem to Computer Science. [online] Medium. Available at: https://fand.medium.com/from-hilbert-with-love-entscheidungsproblem-to-computer-science-c7b35f85384a [Accessed 20 Mar. 2023].

[3] Turing machine pictures from https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html

[4] Shtetl-Optimized. (2011). Rosser's Theorem via Turing machines. [online] Available at: https://scottaaronson.blog/?p=710.

Heavily influenced by:

Wikipedia Contributors (2019). Russell's paradox. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Russell

to, C. (2016). concept in set theory. [online] Wikipedia.org. Available at: https://simple.wikipedia.org/wiki/Cantor

Hodges, A. (1985). Alan Turing: the Enigma. Penguin Random House.

Penrose, R. (1989). The emperor's new mind : concerning computers, minds, and the laws of physics. Oxford ; New York: Oxford University Press.