To Be, or Not to Be? The History of Undecidability in Mathematics

Neil Mehta

Abstract

Undecidability in mathematics has been a longstanding puzzle, intriguing scholars for centuries and reshaping the boundaries of mathematical inquiry. This essay traces the historical trajectory of undecidability, beginning with Cantor's seminal work on the sizes of infinities and its trailing controversy. It then explores the ambitious goals of Hilbert's Program, which aimed to establish a complete and consistent foundation for mathematics but was ultimately challenged by Godel's Incompleteness theorems, revealing the inherent limitations of formal systems. Finally, the essay ends by exploring the formalization and proof of the most famous undecidable problem: the Halting Problem and discussing other contemporary examples such as Conway's Game of Life. Through these historical and modern examples, the essay illuminates the profound implications of undecidability in mathematics and its broader impact on the philosophy of mathematics and computer science.

1 Introduction

The realm of mathematical undecidability deals with questions that are inherently unsolvable or unanswerable by any algorithm or set of rules. More specifically, the definition of undecidability is the non-existence of an algorithm or the impossibility of proving or disproving a statement within a **formal system**.

Definition 1.1. Formal System: An abstract structure and formalization of an axiomatic system used for inferring theorems from axioms by a set of inference rules.

In other words, a formal system (in a mathematical context) is a pair of objects: firstly, a language consisting of symbols from a defined alphabet, governed by a set of grammatical rules; and the second being a deductive or proof system that uses rules of inference that take axioms as inputs and output (or rather, deduce) theorems. An example of a formal system would be the mathematical Field defined in MAT102:

Definition 1.2. Field (MAT102): A field is any set F that is closed under the operations addition $+$ and multiplication $\times$, such that for any $x, y, z \in F$, the following **field axioms** are satisfied:

1. Associativity: $x + (y + z) = (x + y) + z$ and $x \times (y \times z) = (x \times y) \times z$
2. Commutativity: $x + y = y + x$ and $x \times y = y \times x$
3. Identity: $\exists\, 0, 1 \in F$ such that $x + 0 = x$ and $x \times 1 = x$
4. Additive Inverses: $\forall x \in F,\ \exists\, a \in F$ such that $x + a = 0$
5. Multiplicative Inverses: $\forall x \in F,\ \exists\, a \in F$ such that $x \times a = 1$
6. Distributivity: $x \times (y + z) = (x \times y) + (x \times z)$

With the definition of a field, there is a clear language in use that follows a specific set of grammatical rules, and moreover a structure of axioms exist that allow the deduction of consequential results that follow from these core axioms - for example, one claims that if F is a field and $x, y \in F$ satisfy $xy = 0$, then either $x = 0$ or $y = 0$. While the proof is omitted here, it is easy enough to see that the result follows from a manipulation of the above axioms - thus, the field acts as a formal system. If a statement can be formulated within the rules of a formal system but cannot be proven or disproven by the system's rules, it is considered an undecidable problem.

The existence of undecidable problems has profound implications for the foundations of mathematics and computer science. It challenges the notion of algorithmic computability and highlights the limitations of formal systems in capturing the full complexity of mathematical and computational phenomena. In the following sections, the history of undecidability will be explored, tracing its origins from Cantor's work on set theory, to Hilbert's attempt to clarify a foundational basis of mathematics, followed by Godel's

Incompleteness theorems and beyond, shedding light on the evolution of our understanding of mathematical reasoning and computation.

The essay will conclude by examining perhaps the most well-known example of an undecidable problem, the Halting Problem, and discussing other more recent examples such as Conway's Game of Life. The Halting Problem asks whether a given program, when run with a specific input, will eventually halt (finish running) or will run indefinitely. Turing's proof of the undecidability of the Halting Problem is a cornerstone of theoretical computer science, demonstrating the existence of problems that are beyond the reach of algorithmic solutions.

2 History

In order to understand how the notion of undecidability and undecidable problems arose, first the history of formal systems must be examined. In the late 19th century, mathematician Georg Cantor pioneered the field of Set Theory and proclaimed the existence of distinctly-sized infinities to the world with his famous diagonalization argument. At the time, to say his findings were controversial would be an understatement. His discoveries, in effect, had created large cracks in the foundations of mathematics and had sparked a war amongst the worlds greatest mathematicians over the concept of infinity, with Cantor's reputation being greatly harmed as a result.

Theorem 2.1. *The set of real numbers $\mathbb{R}$ is uncountably infinite.*

Proof

We will use this example to illustrate Cantor's proof via his diagonalization argument.

Suppose that $f : \mathbb{N} \to [0, 1]$ is any function. We construct a table of values of f, where each row contains the decimal expansion of $f(n)$. Since $\mathbb{N}$ is countably infinite, the number of rows in the table is also infinite. Suppose $f(1) = e/10$, $f(2) = \sqrt{2}/2$, $f(3) = 7/9$, and $f(4) = \pi/\pi^2$ so the start of the table is as follows:

n	$f(n)$
1	0 . 2 7 1 8 2 8. . .
2	0 . 7 0 7 1 0 7. . .
3	0 . 7 7 7 7 7 7. . .
4	0 . 3 1 8 3 0 9. . .
$\vdots$	$\vdots$

We show that f cannot be surjective by determining an element of $[0, 1]$ that has no mapping from $\mathbb{N}$.

Highlighting the digits along the main diagonal of the table:

n	$f(n)$
1	0 . **2** 7 1 8 2 8. . .
2	0 . 7 **0** 7 1 0 7. . .
3	0 . 7 7 **7** 7 7 7. . .
4	0 . 3 1 8 **3** 0 9. . .
$\vdots$	$\vdots$

The highlighted digits combined into a new number yield 0.2073. . . Suppose we add 1 to each of these digits. The new number obtained is 0.3184. . . We claim that this number appears nowhere in the table. Indeed, this is true because:

- it differs from $f(1)$'s first digit
- it differs from $f(2)$'s second digit
- . . .

- it differs from $f(n)$'s n-th digit

So it cannot equal $f(n)$ for any $n \in \mathbb{N}$. In other words, the number does not appear anywhere in the table. Hence no surjection exists between $\mathbb{N}$ and $[0, 1]$, which implies that there is no bijection between the two sets either, thus they do not share the same cardinality even though both sets are infinite.

Since $[0, 1] \subseteq \mathbb{R}$, we can also say that the same results apply to $\mathbb{R}$ - since no bijection can be formed between $\mathbb{R}$ and $\mathbb{N}$, the two sets have different-sized infinite cardinalities. In particular, $|\mathbb{R}| > |\mathbb{N}|$. $\square$

Henri Poincaré, a prominent French mathematician at the time, deemed Cantor's ideas as a "grave disease" that had infected mathematics. Cantor's former mentor, Leopold Kronecker, denounced Cantor, labelling him a "scientific charlatan" and "corrupter of youth". Others, such as David Hilbert, felt otherwise on Cantor's work:

> This theory is, I think, the finest product of mathematical genius and one of the supreme achievements of purely intellectual human activity.

Mathematicians who rejected Cantor's notions were known as "intuitionists", who believed that "the only real mathematical objects are those that can be experienced", and "mathematical proofs are assertions of the reality of mathematical objects". Those that sided with Cantor and Hilbert were known as "formalists", and attempted "to reduce mathematical problems to formal statements, and then prove that the resulting formal systems are complete and consistent".

The formalists then, led by Hilbert, sought to create a solution to the foundational crisis Cantor's findings had caused in the world of mathematics. Thus, in the 1920's, Hilbert's Program was established - an attempt to create a formal system that was **complete, consistent,** and **decidable**.

Definition 2.2. Completeness: The characteristic of a formal system wherein every true statement is sustained by a valid, rigorous proof.

Definition 2.3. Consistent: The characteristic of a formal system wherein every provable statement is free of contradiction (you cannot prove both A and $\neg A$).

Definition 2.4. Decidable: The characteristic of a formal system wherein an algorithm exists to show whether a statement follows from the system's axioms.

Hilbert was adamant that a formal system satisfying all three requirements did exist, and in 1930 he gave a speech at a meeting of the Society of German Scientists and Physicians in Konigsberg on the topic. However, even at the time of his speech, his search for such a system was under threat. One day earlier, mathematician Kurt Gödel claimed he had found the answer to Hilbert's first condition.

Gödel, born in 1906, was a prominent mathematician, focusing his research in the area of logic. He is regarded as one of the most crucial logicians of all time, with his discoveries being utilized in various proofs even to this day. He is most famous for his two incompleteness theorems, which are widely accepted evidence showing that Hilbert's Program to find a complete and consistent formal system for mathematics is not possible. It was at the same Society of German Scientists and Physicians meeting that Gödel delivered a speech on his incompleteness theorems, the first of which is stated below:

Definition 2.5. Gödel's First Incompleteness Theorem: Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete; i.e., there are statements in the system F which can be factually true, yet have no proof to sustain it.

The proof of the First Incompleteness Theorem requires some preliminary definitions and setup. To prove the first incompleteness theorem, Gödel utilized a system wherein the "provability" of a statement in the language F could be expressed purely in terms of arithmetic functions. These functions could then operate on **Gödel numbers** of statements in the system. As a result, the system, which could prove certain facts about numbers, could then also indirectly prove facts about statements created using the same system.

Definition 2.6. Gödel Numbers: In mathematical logic, a Gödel numbering is a function that assigns to each symbol and well-formed (ie., valid) formula or statement of some formal language a unique natural number, called its Gödel number.

A Gödel numbering can be interpreted as an encoding in which a natural number is assigned to each symbol of a mathematical language. Consequently, a sequence of natural numbers then represents a sequence of symbols forming a statement in the language. These sequences of natural numbers can again be represented by single natural numbers, allowing for their usage in formal theories.

There are three essential steps to the proof by contradiction of the First Incompleteness Theorem:

1. Statements in the formal system F can be represented by a Gödel number. The properties of the statements are equivalent to determining if their Gödel numbers possess certain properties, and that properties of the statements can therefore be demonstrated by examining their Gödel numbers. This step culminates in the summarized idea "S is provable in F", where S is any valid statement in the system F.

2. In the formal system, it is possible to establish a Gödel number G whose statement equivalent is self-referential - in essence, the statement itself says it is disprovable. Aside: this step utilizes a technique called diagonalization, named aptly due to its usage in Cantor's diagonalization argument.

3. This statement can be demonstrated to be neither provable nor disprovable in F, therefore it is impossible to prove the consistency of a formal system's own axioms. Moreover, it is possible to construct a factually true statement, but have no algorithm or proof for the statement - therefore the system F is necessarily incomplete.

Example Formal System with Gödel Numbering

For simplicity, we assume that the language of our formal system T is composed of exactly the following 15 symbols:

- A constant 0 representing zero.
- Two binary functions $+$ and $\times$ representing normal addition and multiplication.
- Three symbols representing logical and ($\wedge$), logical or ($\vee$), and negation ($\neg$).
- Two symbols representing singular and global quantifiers (ie., the symbols "there exists" and "for all" respectively) $\exists$ and $\forall$.
- Two symbols for binary relations, $=$ and $<$ (equality and "less than", respectively).
- Two symbols for left and right parentheses (and) for establishing order of operations.
- A variable symbol x and a distinguishing symbol $*$ to construct additional variables of the form $x^*, x^{**}, \ldots$
- A unary function symbol $\mathbf{S}$ for the successor operation: $\mathbf{S}(n) = n + 1,\ n \in \mathbb{N}$

A well-formed formula is a sequence constructed using these symbols in such a way that when reading the statement equivalent, the statement is well-defined: it makes sense. Thus, for example, $x = \mathbf{SS}0$ is well-formed, while $x = 0 + ++$ is not. In such a system, a theory is defined as a set of well-formed formulas without independent variables.

As a reminder, a theory or statement is consistent if there is no formula F such that both F and $\neg F$ are provable. We now assign each symbol of T a Gödel number. We will use the same encoding as in the book *Gödel, Escher, Bach* by mathematician Douglas Hofstadter:

Number	Symbol	Meaning
666	0	zero
123	$\mathbf{S}$	successor function
111	$=$	equality
212	$<$	less than
112	$+$	addition
236	$\times$	multiplication
362	(	left parenthesis
323	)	right parenthesis

Number	Symbol	Meaning
262	x	variable
163	$*$	star (new variable assignment)
333	$\exists$	singular quantifier
626	$\forall$	global quantifier
161	$\wedge$	logical and
616	$\vee$	logical or
223	$\neg$	negation

Note that by design, no Gödel number of any symbol contains a 0. This is done because when writing out the Gödel number for a formula, each symbol's Gödel number is separated by a 0. This way, any formula can be deciphered from it's Gödel number.

Since each natural number can be obtained by applying **S** to the symbol 0 a finite number of times, every natural number has its own Gödel number, for example $3 = SSS0$ then has Gödel number 123 0 123 0 123 0 666 (note that spaces have been added to the left and right of each 0 for readability - actual Gödel numbers do not contain these whitespaces).

It is obvious that distinct Gödel numbers can be assigned to a finite list of well-formed formulas.

Example: Gödel Numbering of Statements

What follows are examples of valid statements and their equivalent Gödel numbers:

- $x = 1:$ 262 0 111 0 123 0 666
- $1 + 0 = 1:$ 123 0 666 0 112 0 666 0 111 0 123 0 666
- $\exists\, x\, \forall\, x^*\ x^* < x:$ 333 0 262 0 626 0 262 0 163 0 262 0 163 0 212 0 262

Ultimately, Gödel's findings had laid the groundwork for the nearly complete destruction of Hilbert's Program - there is no formal system which is complete over all of mathematics, and it is impossible to prove that the system is consistent within its own ruleset. Hilbert's only solace remained with the question of decidability within a formal system, which would be answered a few years later by none other than Alan Turing.

Alan Turing was a mathematician and computer scientist from England. Born in 1912, he attended King's College in Cambridge where he graduated with a degree in Mathematics. It was at this school where he became a fellow by publishing a proof that some purely mathematical decision problems (problems whose answers are either yes or no) are undecidable. More specifically, he proved that there is no single algorithm that correctly answers "yes" or "no" to every instance of a set of decision problems. In his own words, Turing stated:

> ...what I shall prove is quite different from the well-known results of Gödel ... I shall now show that there is no general method which tells whether a given formula U is provable in K.

However in order to prove his argument, Turing had to, quite literally, invent the modern computer.

Today, computers are a bundle of microchips installed onto a singular circuit-board, paired with some memory and storage and a bright LED screen encapsulated by a plastic or glass container. These computers are incredibly small and incredibly fast; most people carry them around in their pockets and use them absent-mindedly for even the most routine tasks, because they have become so ingrained within human culture.

But this was not always the case. Computing has existed long before transistors were invented. Ancient Greek mathematicians formulated methods to calculate the size and shape of Earth. Pi was approximated as 3.125 by the ancient Babylonians almost 4000 years ago. Medieval merchants planned routes and expeditions around the world, some that were accurate to within a day's length. Humans have been computing for thousands of years, arithmetic being a skill taught just like reading or writing and passed down from generation to generation. Up until the years around World War II, computers were just floors of people performing endless calculations at desks.

It was around this time that Turing, who was obsessed with machinery and mechanical objects his entire life, developed the notion of a "computing machine" - a machine capable of performing the same computations humans do by hand. On the 28th of May, 1936, Turing published a paper titled "On Computable Numbers, with an Application to the Entscheidungsproblem". This paper is regarded as one of, if not the most, famous and important works in computability. In the essay, he rewrote Gödel's arithmetic-based numbering system with formal, hypothetical devices that are now known as **Turing machines**:

Definition 2.7. Turing Machine: A hypothetical, idealised model of a computer's central processing unit (CPU) (CPUs are to computers what the brain is to a human) that controls all computations performed by a computer that uses sequential memory to store data. The sequential memory in a Turing machine is represented by a roll of tape of infinite length, which the machine can use to perform read and write operations.

Sequential memory can be pictured mentally as storing bits of data ontop of each other - on the ground, the last bit of data is stored. The second-last bit is stacked on top of the last, the third-last on top of the second-last, etc. If access to the last bit is required, the computer must traverse its memory from the very top of the stack, the first bit, to the second, the third, to the last one all the way at the very bottom.

In his proof to show that some decision problems are undecidable, Turing discovered the most famous example of an undecidable problem - the Halting Problem.

3 The Halting Problem

In the modern day, many consider algorithms to be computer programs, with the two terms being used interchangeably. During Turing's time, an algorithm was something he considered to be a a list of instructions a human could follow and execute. For an algorithm to solve a problem, it must output the correct solution for every input problem. The Halting Problem is an example of a **decision problem**:

Definition 3.1. Decision Problem: A computational problem whose solution is either "yes" or "no". A decision problem is considered decidable if and only if there exists an algorithm S that solves the problem. Likewise, it is considered undecidable if no such algorithm exists.

An example of a decision problem is "Given a positive integer n, is n prime?". This problem is decidable, as proved below:

Proof

It is enough to provide an algorithm that always outputs the correct answer to show that this problem is decidable. Let S represent this algorithm.
We define S using the following pseudo-code:

```
for some integer i in the range (2, n-1):
    does i divide n?
        then NO, n is not prime
otherwise, YES, n is prime
```

Since S correctly classifies every $n \in \mathbb{Z}^+$, the problem is therefore decidable. □

Definition 3.2. Halting Problem: The Halting Problem is a decision problem that asks "Given a program i and an input x, does i halt on x? Where "halting" essentially means "stopping" or "terminating". In essence, the problem asks whether a given program will end or stop when given an certain input. In convention, the problem is expressed as the *halting set*:

$$K = \{(i, x) \mid \text{program } i \text{ halts when run on input } x\}$$

Theorem 3.3. ***The Halting Theorem**: There does not exist an algorithm S that solves the Halting problem for every program i and input x.*

Proof

Suppose for contradiction that we have an algorithm S that takes a program i and input x, and answers "yes" if i halts on x and "no" otherwise. Note that we make no assumptions on the types of input expected, hence the input x to program i could itself be another program.

Given S, we construct a new program U takes a single input program i, and tests whether i halts when the input data is a copy of itself:

```
DEFINE U(i):
    if S(i, i) answers "yes", then:
        answer "yes"
    otherwise:
        answer "no"
```

We can also construct another program Q that takes the same i, but does the **opposite** of U:

```
DEFINE Q(i):
    if U(i) answers "yes", then:
        loop forever
    otherwise:
        halt
```

We now trace through the call $Q(Q)$:

- Just as in composition of functions, the input of the outer function is whatever the output of the inner function is. In this case, the function composition is $Q(U(S(Q,Q)))$.
- If Q halts when given Q as input, or in other words, if $S(Q,Q)$ answers "yes", then the call $Q(Q)$ runs forever by our function definition. But this means that the program **does not** halt, so the answer should be "no" - this is a contradiction.
- If Q does not halt when given Q as input (ie., $S(Q,Q)$ answers "no"), then $Q(Q)$ **does** halt - another contradiction!

In either case, $Q(Q)$ causes a contradiction, meaning it can neither halt nor continue looping forever - this means our original assumption, the existence of S, must be false - therefore no algorithm exists that can solve the Halting problem for every program i and input x. $\square$

And with this proof Turing shattered what remained of Hilbert's hope for a complete, consistent, and decidable formal system of mathematics. He had shown that there is no one algorithm (in computer science) or proof (in mathematics) which could correctly answer or solve every statement within a single system. With his proof that the Halting Problem is undecidable, Turing effectively showed that computing machines will always have a set of problems which can never be solved - this result holds even for modern-day machines. For most, such an outcome is all but inconsequential.

But in the realm of computer scientists, Turing's discovery has several important implications. The undecidability of the halting problem lets computer scientists reason about problems that are solvable, and problems that are virtually impossible. Understanding the limits of the computability allows for making reasonable assumptions with respect to certain problems, as well as accepting that written code will always contain imperfections. As a result of these findings, computability has since become a significant topic of research in the area of computability theory in mathematics, still growing to this day. A broad overview of real life applications include computer security and threat detection, error handling, even smartphone applications are a version of the Halting Problem. There are a variety of (relatively) modern examples of other undecidable problems, such as Conway's Game of Life, which simulates a particular version of a Turing machine.

4 Conway's Game of Life

Conway's Game of Life, also known simply as Life, invented by the British mathematician John Horton Conway in 1970, is an example of a **cellular automaton**—a grid of cells that evolves over discrete time steps based on simple rules. Despite its simple definition, the Game of Life exhibits complex and unpredictable behavior, making it a fascinating subject of study in mathematics and computer science.

Definition 4.1. Cellular Automaton: A discrete model of the dynamic behavior of a system consisting of a grid of cells, where each cell changes its state over discrete time steps according to a set of rules. It consists of a regular grid of cells, each in one of a finite number of states, such as on and off (in the simplest case). The grid can be in any finite number of dimensions.

For each cell in a cellular automaton, a set of cells called its neighborhood is defined relative to the specified cell. An initial state (time $t = 0$) is selected by assigning a state for each cell. A new generation is created by advancing t by 1, according to some fixed rule (generally, a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously.

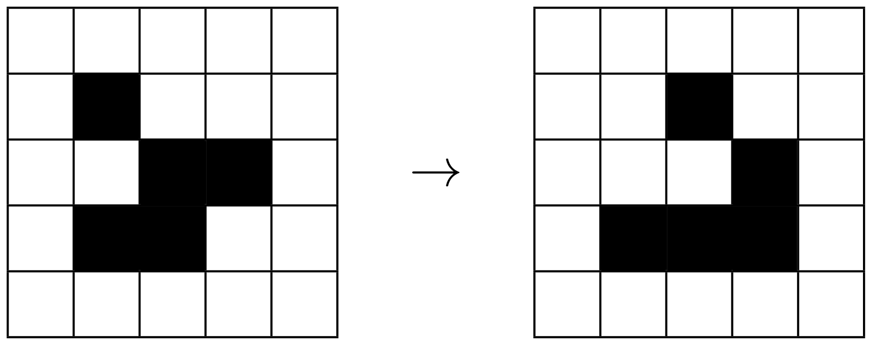
John Horton Conway, born in 1937, was a prolific mathematician known for his contributions to various fields, including group theory, number theory, and combinatorial game theory. He was inspired to create Life after reading about the work of mathematician John von Neumann, who had proposed a self-replicating machine based on cellular automata. Conway wanted to design a game with simple rules that could produce complex and lifelike patterns, akin to the growth and evolution of living organisms.

Life is played on a 2-dimensional **infinite grid** of cells. Each cell possesses one of two states: alive or dead, and eight neighbours: the cells that are horizontally, vertically, and/or diagonally adjacent. The game evolves by generations based on the following rules:

1. Any live cell with strictly fewer than two live neighbours dies, as if by underpopulation.
2. Any live cell with exactly two or three neighbours lives on to the next generation.
3. Any live cell with strictly greater than three neighbours dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbours becomes live, as if by reproduction.

The initial pattern is known as the *seed* of the system. Below is an example of a seed evolution over one generation, also known as a *tick*. Gray cells are live, while white cells are dead:

Example Seed Evolution for 1 Tick:



Theorem 4.2. *The Game of Life is an undecidable problem.*

What follows is a sketch proof of the above theorem. Proving that the Game of Life is undecidable involves showing that there is no algorithm that can determine whether a given configuration of the Game of Life will eventually stabilize (i.e., reach a stable state where no further changes occur) or continue indefinitely.

To do this, we can reduce the Halting Problem to the Game of Life, which means showing that we can construct a configuration in the Game of Life that simulates the behavior of a Turing machine, such that the Turing machine halts if and only if the Game of Life configuration stabilizes.

The key idea is to encode the tape of a Turing machine and its current state in the configuration of the Game of Life. We can then define rules for the Game of Life that simulate the behavior of the Turing machine, with certain configurations corresponding to the halting states of the Turing machine.

Since the Halting Problem is undecidable, if we could determine whether the Game of Life configuration stabilizes, we could also solve the Halting Problem, which is a contradiction. Therefore, the Game of Life is undecidable.

Life is an example of an undecidable problem being investigated for stabilizing algorithms even to this day - breakthroughs and new discoveries of recurring cellular patterns have occurred as recently as 2018. The game has served as an inspiration for the growing field of biological simulation. Life simulations provide judgment on the strengths of competing hypotheses, and generate unexpected or unsuspected possibilities for biologists to study and prove empirically, which highlights the importance of its development back in 1970.

5 Conclusion

The concept of undecidability, central to mathematical theory, denotes questions that defy resolution by any algorithm or formal system. Specifically, undecidability signifies the absence of an algorithm or the inability to prove or disprove a statement within a formal system. When a statement can be formulated within the rules of a formal system but remains unproven or unrefuted by those same rules, it enters the realm of undecidable problems.

Mathematical undecidability's implications reverberate through the foundations of mathematics and computer science, challenging the notion of algorithmic computability and exposing formal systems' limitations in capturing the intricacies of mathematical and computational phenomena. We have traversed the tumultuous and eventful history of undecidability, from Cantor's set theory causing fractures to the very foundations of mathematics, to Hilbert's quest for a complete, consistent, and decidable mathematical system being derailed by the findings of Gödel through his Incompleteness theorems and Turing's Halting Problem. Hilbert died in 1943, with a phrase from his 1930 speech mentioned earlier engraved on his tombstone, bolded below:

> We ought not believe those who today, with a philosophical air and reflective tone, prophesy the decline of culture, and are pleased with themselves in their own ignorance . . . Instead of this silly ignorance, on the contrary let our fate be: "**We must know, we will know**".

The truth is, sometimes we can't know. But that doesn't stop us from trying. Throughout history, humans have strived to unlock the secrets of mathematics. Without Alan Turing's Turing machines, there would be no modern-day computer. Without Gödel's Incompleteness Theorems, there would be no Turing machines. Without Hilbert's Program, there would be no Incompleteness Theorems. Without Cantor's Diagonalization Argument, there would be no Hilbert's Program.

The entire history of mathematics stems from bending the rules until they break, from proving what was once taken as ground truth to be false. One day, a solution to the Halting Problem may arise with the development of new methods and theories, for example through quantum computing. Only through the unearthing of cracks can a stronger foundation be built, the glory of discovery is only born from the ashes of contradiction.

6 Bibliography

Gardner, Martin. “MATHEMATICAL GAMES.” Scientific American, vol. 223, no. 4, 1970, pp. 120–23. JSTOR, http://www.jstor.org/stable/24927642. Accessed 29 Mar. 2024.

Hilbert, David. “On the Infinite.” Philosophy of Mathematics: Selected Readings. Ed. Paul Benacerraf and Hilary Putnam. Cambridge: Cambridge University Press, 1984. 183–201. Print.

Raatikainen, Panu. “Gödel’s Incompleteness Theorems.” Stanford Encyclopedia of Philosophy, Stanford University, 2 Apr. 2020, plato.stanford.edu/entries/goedel-incompleteness/#FirIncThe.

Translation Of An Address Given By David Hilbert In Konigsberg, Fall 1930. Address given by David Hilbert, 1930. (n.d.). https://mathweb.ucsd.edu/ williams/motiv/hilbert.html

Turing, A.M. (1937), On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, s2-42: 230-265. https://doi.org/10.1112/plms/s2-42.1.230