*The Plymouth Student Scientist, 2021, **14**, (1), 405-451*

Mathematical constraints and their philosophical impact

Adam Stiles

Project Advisor: Daniel Robertz, School of School of Engineering, Computing and Mathematics, University of Plymouth, Drake Circus, Plymouth, PL4 8AA

Abstract

This paper investigates some of the constraints encountered in mathematics. It proves the incompleteness theorems. It investigates or discusses theorem proving procedures, complexity, undecidability, undefinability, and covers some examples of independent formulas. Some dialogue on the influence of these concepts on the development of mathematics, and on mathematical philosophy since the 20^{th} century is included. This paper should give any person with an interest in philosophy sufficient understanding to discuss mathematical philosophy with mathematicians.

Keywords: mathematical constraints, incompleteness theorems, complexity, undecidability, undefinability, independent formulas, mathematical philosophy

Introduction

This paper is designed to be understood by both mathematicians and non-mathematicians. It includes considerably more words then would be found in a textbook dealing with a similar number and depth of topics, and has frequent translations between symbolic logic and prose. A glossary of notation is included for reference. All primary sections are self-sufficient if read by someone already acquainted with logic or mathematical writing.

I will omit some proofs of theorems I have used, especially where the proof is available in the literature and the theorem is well known. Wherever possible I will provide a reference to another mathematician's proof. Basic operations which do not provide insight, such as multiplication and exponents will not be defined explicitly. Some operations and definitions, although understood by both the author and the reader will not become available until after certain points, such as addition not being defined until after Peano Arithmetic. I will assume that the reader is familiar with a certain amount of propositional logic, set theory and is used to computers. However, most readers should be able to understand the majority of the paper without these.

Notation becomes slightly tricky sometimes since we have to make the method and maths we are using clear which involves using the normal symbolic language we can understand. At the same time, we have used these methods to manipulate symbols in a way which might not be consistent with their normal syntactic meaning (We will see unmatched parenthesis later, for example).

A very informal way of explaining this use of symbols and language would be that we have two languages, one for explaining and proving things about the other language, the second being the 'true' internal language of the maths. We shall almost always be manipulating symbols in the language of explanation and proof, rather than the language of what we are proving.

We will cover a section of mathematics deeper than it is broad. At many steps, there will be different choices to make about what to cover in detail; and the alternative choices are equally valid. As a result, we will mention some other relevant mathematics that was not covered, and that could have been used instead to emphasize the fact that the conclusions can come from more than one place.

Language is mentioned only briefly, and this sets the tone of the centrepiece of this discussion since it leads to proving incompleteness over undefinability. Peano arith-

metic is used as our number system rather than Robinson or recursive arithmetic since it is the system of numbers the reader is most likely to be familiar with, although the incompleteness theorems can be proven in a weaker arithmetic system.

We discuss the resolution calculus rather than the sequent calculus (another systematic theorem proving procedure) since the resolution calculus is simpler and this makes it a better example. Additionally, even after choosing to use the resolution calculus, we will choose to omit the variant dealing with formula in a first order language, since it would add difficulty without additional meaning.

We will go through the proof of the incompleteness theorems from the ground up, and that is the central point of this work. It is to be hoped that seeing a long, involved proof of this kind will help a reader to interpret proofs and theorems of a similar kind, as well as the importance of very precise nuances in mathematics.

The descriptions of the axiom of choice and the continuum hypothesis support the purpose of this work, by showing some of the constraints on pure mathematics before we move on to the constraints computers face. A great deal of the theoretical part of this last section involves the apparent difference between solving a problem and verifying the solution to one, which is another constraint of interest.

The Philosophical Context

This section is inspired by and uses information from [13] and [16].

Mathematical philosophy is an unusual subject, and one which is not obviously important since the majority of mathematicians will continue to do maths regardless of which (if any) of the philosophical theories turns out to be true; and that assumes that we can found out which of the theories is true, which seems unlikely. Even so, it is worth talking about, if only because it gives us some idea of the limitations of mathematics before we even start to prove a theorem. It will also emphasize the enormous impact that the incompleteness theorems had, and to a lesser extent, the impact computers have had on the world of mathematics.

As with all philosophy, there are a large number of different theories, and variants which have been created as a result of those theories; and those who endorse those variants may or may not object to their theory being called a variant. Even so, we can split most mathematical philosophers into two groups: Platonists and anti-Platonists. This kind

of dichotomy is present in most of philosophy, under different names depending on the discipline; we have empiricists and rationalists, dualists and physicalists, and cognitivists and non-cognitivists. Many of these positions are analogous to Platonism and anti-Platonism in different topics, but this paper will not try to build these comparisons.

Platonists follow Plato (and presumably Socrates) in asserting that mathematical objects such as numbers, shapes and theorems exist independently of our perception of them. They also claim that these mathematical objects inhabit some purer universe and that human brains (Plato would have said souls) have some ability allowing us to recognise mathematical objects from this world, and apply them to real life. Plato himself tried to construct a proof of this in [26], but it is not convincing. Despite this, Platonism has a following among mathematicians because it gives real-world meaning to abstract mathematical truth.

Anti-Platonists all reject the notion that mathematical objects inhabit some 'purer' world and exist outside our perception of them, but the different anti-Platonist theories all disagree about what exactly mathematical objects are. I will use the word 'good' as a way of describing the philosophies' views on what mathematics is not nonsense. Logicists state that all 'good' mathematics can be reduced to logic, statements in a zeroth order language (precisely what this means is explained later). Predicativism is similar to Logicism but claims that all 'good' mathematics can be reduced to statements in a first order language. Intuitionists claim that the basis of mathematics is human intuition and that if humans cannot understand something intuitively, it cannot be 'good' mathematics.

There are fairly strong objections to these three anti-platonist viewpoints. Logicists have never been able to reduce mathematics to logic, despite several attempts, and so it may not be possible at all. The same is true for predicativists, they have not yet been able to reduce higher mathematics to first order mathematics. Succeeding in these tasks is a prerequisite for entering the stage of meaningful philosophical discussion, and that is beyond the scope of this paper. Higher mathematics cannot be dismissed as 'too abstract' to be worth studying either; for example, Maxwell's equations must be in a second order language. One of the implications of intuitionism is that it disallows proofs of the existence of an object which do not explicitly find the object in question, and they reject some discussion of infinity. Infinities are common in all kinds of mathematics, including the definitions of confidence intervals and Taylor expansions. As a result, we will not consider intuitionism either since accepting it results in mathematics very different to that the reader is likely to have encountered, and this is not useful. The final significant anti-platonist theory is formalism.

Formalism follows from Kant's teachings, but its modern form is mostly Hilbert's creation. Formalists assert that the study of mathematics is simply the investigation of different axiomatic systems; and that none of these systems is any more real than any other. They claim that mathematical objects are only true if they are direct results of these axioms. and are only ever true while we work in that system. For example, a formalist would claim that the study of mechanics is not an attempt to learn how real objects act, but simply an investigation into the problems when operating in a specific system of rules. This is the most practical and simplest view, and Occam's Razor would certainly support it. However, it is unappealing to some mathematicians since it makes mathematics seem somewhat pointless, and because mathematical objects and structures do seem to exist in the real world; stars are (approximately) spherical.

This paper treats formalists as the strongest anti-platonist theory and considers their views against the platonists', since it is useful to have a concrete representative for that side of the dichotomy, and formalism is an acceptable contender for platonism. They both have major flaws and advantages, and I will illustrate the reaction of these groups to the major theorems I prove and mention.

It may seem odd to any members of the more practical sciences that we refuse to reject an unnecessarily complex viewpoint (Platonism) just because we like to believe in it, but in mathematics and philosophy this is not unusual. Philosophers of the mind frequently reject theories which lead to not having free will; mathematicians construct axiom systems in which specific things happen just so those things happen. It should not be surprising then, that the philosophy of maths also has this property.

Unsolved and Unsolvable

Much of the information on Hilbert's and the Millenium problems was obtained from [14] [15] [29].

It is important to note the distinction between mathematical problems and formula for which no solution has been discovered; and those problems which have been proven to have no solution. There are many examples of both categories, such as the Goldbach conjecture in the first category, stating that every even number greater then two is the sum of two primes, and the axiom of choice in the second. There are two famous lists of problems which were unsolved when published. These are the Millenium Problems, published in the year 2000, and Hilbert's 23 questions, published 100 years before.

Several of Hilbert's 23 questions have been either solved or shown to be unsolvable. This includes the second incompleteness theorem and the truth of the continuum hypothesis which I discuss in this work. Only one of the 7 Millennium problems have been solved, the Poincaré conjecture. Deciding (or proving the undecidability of) P $\stackrel{?}{=}$ NP is also a Millennium problem.

Elements of Number, Logic and Language

In this section, we will define the systems of logic and number we will use for the rest of this dissertation. This will involve definitions of language and language order, as well as formulae. It will also include the axioms of Peano arithmetic which we will be using, with some reflections on why certain axioms are necessary and the difficulties of defining them in the correct language.

Language

Definition: A **language** is a collection of symbols and a collection of rules about how those symbols can be placed sequentially.

Definition: A **formula** is one symbol, or several placed sequentially. A **well formed formula** is a formula obeying the rules of our language. Unless it is especially relevant, we will treat all formulae as well formed.

All logic and mathematics are described using a language. The *Principia Mathematica* [30] used a different language than the language that is widely used today for the same topic, but neither is incorrect. It is also worth noting that this definition of a language includes both spoken and written (simplified) English, the binary that computers use to communicate, as well as the languages of mathematics and logic.

All languages have an order, a way of describing which objects the language is allowed to make statements about. The three important orders are zeroth, first and second order. Discussion of the rules defining what a zeroth order language can and can't do and theorems about zeroth order languages are known as zeroth order logic, and so on.

Zeroth order languages contain variables, and can define operations between them. For example, $\theta_1 \wedge \theta_2$. It has no quantifiers, such as $\forall$ or $\exists$.

First order languages are zeroth order languages but with the addition of quantifiers. A language of this order allows us to define sets and perform arithmetic. More formally, it "Quantifies variables only over individuals". For example, where ψ is a well formed formula, $\exists x : \psi(x)$ is a statement in a first order language [4] [19].

Second order languages are first order languages which can also quantify variables over properties as well as individuals. This means that we can use quantifiers in conjunction with properties; for example, $\exists M$ such that x is an object with the property M. The set of real numbers has not been defined in less than a second order language since both Dedekind cuts and Cauchy sequences use second order language terms, and neither do any of the other existing definitions of the real numbers. Details about Cauchy sequences and other interpretations of the real numbers can be found in [11].

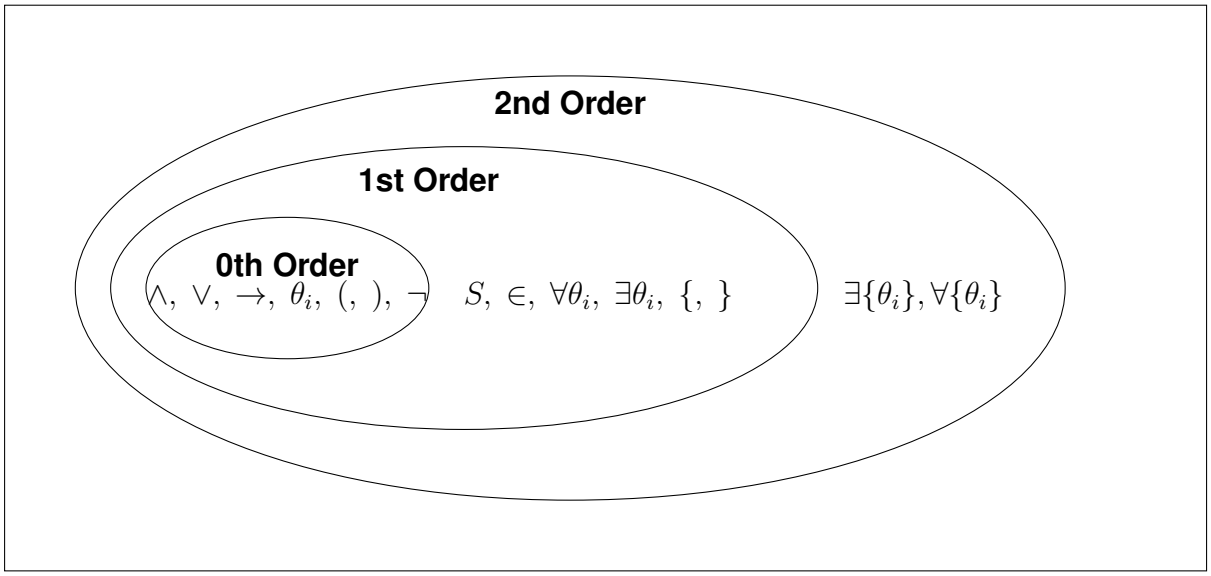

Figure 1: Euler Diagram of Some of the Symbols and Syntax Rules Usable in Each Language Order

Peano Arithmetic

To use ideas like proof by induction and recursive functions, we must define the rules of the system of arithmetic we will use going forwards; Peano Arithmetic (PA). PA is essentially the normal system of arithmetic we use as a basic starting point for a large amount of maths, including calculus, computer science and other real-life applications. It is still interesting to define it, especially since we will not use the standard definition, and attempt to obtain the same system despite this.

PA can be defined using axioms in a first or second order language. We will use a first order language since the incompleteness theorems were originally written using a first order language. It is worth noting that if we define PA using a first order language, that definition will still give us arithmetic even if we are operating in a second order language.

There are other systems of arithmetic, such as Robinson Arithmetic and recursively defined arithmetic, but PA is the most familiar to most people. The most formal way to describe PA is that it is a model which sets the axioms of arithmetic to true, and so we will just state the formula we will set to true as the definition of the model.

To define PA, we will start with an arbitrary set Φ, and then manipulate it by adopting certain axioms, until we can show that we end up with the natural numbers we are familiar with. We will also define a way of ordering this set, which will give us addition. Our first axioms are:

1. Φ is a set
2. $1 \in \Phi$
3. $\forall a, b \in \Phi,\ a = b$ is a symmetric, reflexive and transitive relation.
4. If $b = c$ and $b \in \Phi$ then $c \in \Phi$

At this point, Φ does not look much like the natural set. To fix this, we will define a successor function S, which will enable us to order this set. This will take the role of 'adding 1' in normal arithmetic; but without performing addition, since that would imply that we already knew what addition meant. We do know what a function is, as this can be defined without arithmetic (although we have not defined a function explicitly). Assuming we knew what addition meant would be out of place, given that we haven't yet defined what the natural numbers are. The second collection of axioms are:

1. $\forall n \in \Phi,\ S(n) \in \Phi$
2. $\forall n, m \in \Phi,\ m = n$ if and only if $S(m) = S(n)$
3. $\forall n \in \Phi,\ S(n) \neq 1$

Now that we have the successor function, Φ has many of the properties of the natural set, and $\mathbb{N} \subseteq \Phi$. However, there are many other candidates for Φ which contain elements with different properties to the normal counting numbers. We call these additional elements of Φ non-standard numbers. To ensure that we obtain $\Phi = \mathbb{N}$ we must

exclude these elements. (This piece on non-standard numbers was inspired by [31]).

These non-standard numbers can be of two types; they can be cyclical or exist as part of infinitely long chains which do not contain unity and are alternatives to the primary chain of counting numbers. The cyclical non-standard numbers take the form $a, b, c \in \Phi$ and $S(a) = b,\ S(b) = c,\ S(c) = a$. The infinite alternative chain takes the form $(... < a < b < c < d < ...)$ where $a,\ b,\ c,\ d \in \Phi$ but are not equal to any of the counting numbers 1,2,3... etc.

A system of arithmetic containing these non-standard numbers would not be very useful since it would be inelegant to define common and important concepts such as prime numbers on a set containing them. Our set Φ was arbitrary, and it has no rules on what can be in it apart from the axioms we have already defined. So, we must adopt an axiom which ensures that the natural set contains only the normal 'counting' numbers and eliminate a, b, c, d and symbols with the same properties.

The most usual and intuitive axiom to eliminate this problem would be the axiom of induction:

If H is a set such that: $1 \in H$ and $\forall n \in \Phi$, $n \in H \rightarrow S(n) \in H$, then H contains every element in Φ.

However, the axiom of induction is an axiom which makes use of terms only available in a 2nd order language as it quantifies over properties of H. We are restricting ourselves to first order. So instead, we must use an axiom schema (an axiom which contains an infinite number of choices for the key variable in the axiom) to exclude the non-standard numbers. Situations like this are why it is important to specify the order of the language we use.

The axiom schema we will use is:

For every well-formed formula ψ (we will henceforth use ψ for a formula in the language of PA without clarification):

$$[\psi(1) \rightarrow (\forall x_i \in \Phi, \psi(x_i) \rightarrow \psi(S(x_i)))] \rightarrow [\forall x_i, \psi(x_i)]$$

Or in slightly plainer English, if $\psi(1)$ implies that for every element in Φ, $\psi(x_i)$ implies $\psi(x_i + 1)$, then we can conclude as a result of this axiom that for all x_i, $\psi(x_i)$ is true. Non-standard numbers do not obey this axiom, and so they can be excluded from Φ.

[6]

Since we have excluded the non-natural numbers, Φ contains exactly the same elements that $\mathbb{N}$ does; they are equal. From here on, we will refer to $\mathbb{N}$ rather than Φ. The successor function and equality have also been defined, and we have the natural set. This is sufficient to define addition and many other elementary operations and properties on the natural set, such as multiplication, and the divides relation $n|m$. We will now assume we know what these things mean.

If we wanted, we could add additional axioms into PA. There are many reasons why we might want to do this; for example, we might want to define the rational numbers and division. For us, it will come in later when we want to generalise the incompleteness theorems, giving us a theory which applies to systems containing arithmetic.

Formal Logic

In this section, we will introduce formal logic and examine some ways to systematically prove theorems in this system. This is essential for understanding later statements about truth and as an example of complexity.

Formal logic is the study of valid inference and elementary laws of truth, including statements such as: 'either θ_1 or θ_2 is true'. Formal logic allows us to express statements of this kind in symbols and develop systematic ways of looking at them. It is important to us because it is the basis for all operations involving deductions and decisions based on truth, such as mathematics and computing. Also, there are some simple classes of problems in formal logic which become useful in the study of complexity, such as the satisfiability problem.

Formal logic uses a 0th order language. Formal logic is extremely restrictive, since some statements which humans can understand and know the truth of intuitively cannot be expressed in it. One example of this is the syllogism: 'Timmy is a cat. All cats are mammals. Therefore Timmy is a mammal'. However, it is meaningful to have this elementary system if only to show the difficulty of solving certain kinds of problems, since if they problems are hard in this simple system, they will almost certainly grow in difficulty when we begin to use more sophisticated systems.

One constraint of 0th order languages is that they struggle to prove things about them-

selves since it is hard to construct any meaningful general proof without the ability to use induction or quantifiers, which they do not have. For example, the resolution calculus is defined in a first order language but operates only on objects in a 0th order language. We will see later that first order languages are capable of being used to prove things about themselves.

An Example of Formal Logic

One of the purposes of formal logic is to determine whether equalities like the following are true or false.

$$x_1 \wedge (x_2 \rightarrow x_3) \equiv (x_1 \wedge x_2) \rightarrow x_3$$

To determine the truth of this equality, we would need to examine all the permutations for assigning variables. When all variables are false, we can see that the first formula is false and the second formula is true. These do not give the same output for the same model; so they are not equivalent, and the formula as a whole is false.

The Resolution Calculus

This subsection was inspired by questions given to me by my supervisor Daniel Robertz and uses some of his definitions and notation.

The resolution calculus is a systematic way of determining whether a formula in formal logic is satisfiable (there is an assignment which makes the formula output to true) or not. This problem is a useful example in itself, as well as in complexity. There is an extension of the resolution calculus which operates over first order language formulas as well, but it has limitations which will make it a poor example, and so we will restrict ourselves to 0th order formula.

Definition: A **literal** is a propositional variable or its negation. We will use L to represent the general literal.

Definition: A **clause** is a finite set of literals. Let $\square$ be the empty clause.

Definition: A formula is said to be in **conjunctive normal form** if it is of the form:

$$(L_{1,1}, \vee ... \vee L_{1,m_1}) \wedge ... \wedge (L_{n,1}, \vee ... \vee L_{n,m_n})$$

All logical formulas can be represented in conjunctive normal form (CNF). We will not prove this explicitly, but it can be seen by constructing an algorithm which transforms any formula into that form by using a variety of rules and distributive laws.

We assign every formula in CNF a unique set of clauses, such that each of the clauses corresponds uniquely to one of $(L_{i,1}, \vee ... \vee L_{i,j_i})$. Since the assignment is unique, we can use these forms interchangeably.

Now, we can define the resolution calculus. Given two clauses C_1 and C_2, and assuming that there is some $L \in C_1$ and $\neg L \in C_2$, $(C_1 \backslash \{L\}) \cup (C_2 \backslash \{\neg L\})$ is a resolvent of C_1 and C_2. (There can be multiple resolvents created by any two clauses; this is annoying but will be able to deal with this later).

We can make the resolution calculus more useful by formalising an operator for taking the resolution of two clauses. This will make it easier to use this calculus inductively and recursively. We use 0 here despite our natural set starting at 1; this is an example of the difference between the languages of proof and what we are proving things about.

1. Let $Res(K) = K \cup C$, where C is a resolvent of two clauses in K.
2. Let $Res^0(K) = K$
3. Let $Res^{n+1}(K) = Res(Res^n(K))$
4. Let $Res^*(K)$ be the union of all possible repeated resolvents of K

A Lemma Enabling an Application for the Resolution Calculus

Definition: We use $A(\psi) = 1$ to mean that A is a model which makes ψ true, and $A(\psi) = 0$ to mean that A is a model which makes ψ false. I also (somewhat colloquially) treat a formula and the set which represents it as being the same thing. We can do this without loss of rigour since they are defined uniquely.

The following lemma will help us to use the resolution calculus to prove things: **for a set of clauses K and $C_1, C_2 \in K$ and for a resolvent of C_1 and C_2, R, then K and $K \cup \{R\}$ are equivalent.** Equivalence means that they give the same output for each assignment of variables.

Let A be any model such that $A(K) = 1$. We want to show that this implies that $A(K \cup \{R\}) = 1$. We know that any resolvent of C_1 and C_2 is $(C_1 \backslash \{L\}) \cup (C_2 \backslash \{\neg L\})$. The truth output of one of these clauses must be independent of the literal Y or its

negation because of our assumption, and so the union of the two must also be true; since all the operators here are $\vee$, and therefore only one of the clauses needs to be true.

To prove the other direction: let A be any model such that $A(K \cup \{R\}) = 1$. We want to show that this implies that $A(K) = 1$. From the definition of the union and because the formula is in conjunctive normal form, K is a select part of $K \cup \{R\}$ and therefore the model outputs it to true.

Therefore since if either is true the other must be true, and the only other option is that one of them is false which would make the other false, K and $K \cup \{R\}$ are equivalent.

Because the union operator is associative, any union of K and any of its resolvents are equivalent. This means that if we can prove that $Res^*(K)$ is satisfiable or tautologically true, we will know that the same is true for K, which will become a useful technique for us.

An example of the operation of the Resolution Calculus

The following example is intended to show how a human could use the resolution calculus to determine whether a formula in conjunctive normal form is satisfiable. The way a computer would do this is roughly the same, but it would consider every pathway of the resolution calculus, whereas I only consider the one leading directly to the result.

The formula we use for this example is:

$$(\theta_1 \vee \neg\theta_2) \wedge (\neg\theta_3) \wedge (\neg\theta_1 \vee \neg\theta_2 \vee \theta_3) \wedge (\theta_2 \vee \theta_3)$$

This is equivalent to the following set of clauses:

$$\begin{cases} C_1 = \{\theta_1, \neg\theta_2\} \\ C_2 = \{\neg\theta_3\} \\ C_3 = \{\neg\theta_1, \neg\theta_2, \theta_3\} \\ C_4 = \{\theta_2, \theta_3\} \end{cases}$$

Applying the resolution calculus to this twice in a specific way, we find 2 new clauses:

$$\begin{cases} C_5 = \{\neg\theta_2, \theta_3\}\ (From\ C_1, C_3) \\ C_6 = \{\theta_2\}\ (From\ C_2, C_4) \end{cases}$$

Taking the resolution of C_5, C_6, we obtain $C_7 = \{\theta_3\}$. For reasons that will become clear, we can stop here.

From the previous lemma stating that K is equivalent to $K \cup R$ and from the associative property of the union operator, K is equivalent to $K \cup \{C_1\} \cup \{C_2\} \cup ... \cup \{C_7\}$. This union of sets has a equivalent in conjunctive normal form. The logical formula in CNF for $K \cup \{C_1, C_2, ..., C_7\}$ is as follows:

$$(\theta_1 \vee \neg\theta_2) \wedge (\neg\theta_3) \wedge (\neg\theta_1 \vee \neg\theta_2 \vee \theta_3) \wedge (\theta_2 \vee \theta_3) \wedge (\neg\theta_2 \vee \theta_3) \wedge (\theta_2) \wedge (\theta_3)$$

This formula states that θ_3 must be both true and false for the formula to be true. This cannot happen, so the formula is always false; it is unsatisfiable. Since this formula is equivalent to K, K must be unsatisfiable as well.

Using the Resolution Calculus to Prove Satisfiability

The following theorem will enable us to define an algorithm which uses the resolution calculus and can determine for any formula in formal logic, whether that formula is satisfiable or not.

The theorem is: **a set of clauses K is unsatisfiable if and only if $\square \in Res^*(K)$.** A disjunction can only be true if it contains some combination of literals, at least one of which must be true. Therefore, the empty disjunction is always false. This means that any set containing the empty clause is unsatisfiable. In addition, a formula needs some clauses to be false in order to be false. If a formula has no clauses, it is represented by the empty set, and it is true. This is not obvious, but having these rules makes our system easier to work in.

Proving the backwards direction; showing that a set of clauses K is unsatisfiable if $\square \in Res^*(K)$. If the empty clause is a clause in $Res^*(K)$, then $Res^*(K)$ is unsatisfiable. If $Res^*(K)$ is unsatisfiable, then K is also unsatisfiable, from the lemma proved in 5a and the associativity of the union operator. Therefore, K is unsatisfiable if $\square \in Res^*(K)$.

Proving the forward direction; a set of clauses K is unsatisfiable only if $\square \in Res^*(K)$.

Let $x_1, x_2, ..., x_n$ be the full set of variables occurring across all the clauses of K. Let K^+ be the set containing all the clauses of K which do not contain x_n, and remove $\neg x_n$ from all the clauses in K^+. Let K^- be the opposite; the set containing all the clauses of K which do not contain $\neg x_n$, and then remove x_n from all the clauses in K^-.

For this direction of the proof, we must assume that K is unsatisfiable. Assume that the satisfiability of K depends only on x_n; which is to say that for the formula to be

true, x_n would have to be both true and false. If K^+ was then satisfiable, then it would imply that K was satisfiable, since we could copy the assignment for satisfiability in K^+ and add the assignment x_n is true to it. This violates our assumption that K is unsatisfiable. Therefore, K^+ is unsatisfiable; and it is easy to see due to symmetry that K^- must be unsatisfiable too.

Assume that $\square \in Res^*(K^+)$ and $\square \in Res^*(K^-)$.

Let R^+ be a specific collection of clauses used to derive the empty clause in $Res^*(K^+)$. Either some of the clauses in R^+ originally contained $\neg x_n$ before we removed them, or they did not. If they did not, we should return to the step where we define R^+ and substitute x_n for x_{n-1}. We know that there is at least one variable for which the first option will work, because otherwise K^+ would be satisfiable.

In the case where some clauses did contain $\neg x_n$ we can add it back in; and now, the clause that was previously considered empty contains $\neg x_n$, since it cannot be eliminated from $Res^*(K^+)$ as x_n is not present. Therefore, $\{\neg x_n\} \in Res^*(K)$, since $R^+ \subseteq Res^*(K^+)$ and when we add $\neg x_n$ back into the clauses in K^+, we know that $Res^*(K^+) \subseteq Res^*(K)$.

Similarily and under the same assumptions, we can conclude that $\{x_n\} \in Res^*(K)$. The resolvent of $\{x_n\}$ and $\{\neg x_n\}$ is the empty clause.

We have made the critical assumption that $\square \in Res^*(K^+)$ and $\square \in Res^*(K^-)$, and we know that if this is true, then $\square \in Res^*(K)$. We can now use recursion to prove this assumption. We know that K^+ is unsatisfiable. We can then replicate this entire proof on K^+, obtaining the conclusion that if $\square \in Res^*(K^{+2})$ and $\square \in Res^*(K^{-2})$, $Res^*(K)$ contains the empty clause, where K^{+2} and K^{-2} are the new sets created during the proof as constructed earlier.

We can see that this process will continue if K is unsatisfiable until both K^{+i} and K^{-i} contain only empty clauses since the number of the variables in these sets we make assumptions about will gradually decrease. As a result, these assumptions at some point stop being assumptions and become truths as variables are eliminated.

Therefore, K is unsatisfiable if and only if $\square \in Res^*(K)$.

It is relatively easy to prove that for a finite set the number of resolvents that can be taken of that set is also finite; I have omitted this. So if we continuously take resolvents

of a set of clauses, we will either have one of the resolvents be the empty set or not; and then we will know if the original formula is satisfiable or not.

The Incompleteness Theorems

My aim in this chapter is to prove the following theorems:

The first incompleteness theorem: Any recursively axiomatisable and consistent theory that is capable of performing a certain amount of arithmetic is incomplete; there are statements in that theory which are neither provable or disprovable.

The second incompleteness theorem: The consistency of PA cannot be proven inside PA.

These theorems were extremely influential in altering the zeitgeist of mathematics in the 20th century, and they have a wider philosophical importance. However, the mathematics needed to prove them and understanding their rigorous definition is complex enough that they are little known in the wider philosophical community and sometimes misused. The proofs in this section aim to be more accessible then those found usually whilst maintaining rigour.

The flowchart at the end of this section is designed to give a general idea of how the proof of the incompleteness theorems progress, rather than to be rigorous. The circle represents our starting model, the diamonds represent constructions, the rectangles represent proofs, and the rectangles with rounded corners are our conclusions. The arrows show which concept or proof are applied to which other concept or proof; and the arrows are transitive, so that ”Completeness Theorem” and ”Theorems of Arithmetic” are used in the proofs of ”Effective Methods”.

Results of PA

We can obtain many interesting and important theorems from these axioms alone with important applications, such as encryption, but we will limit ourselves to those which will be useful later.

Gödel's Completeness Theorem

Amusingly, one of the proofs on the way to prove Gödel's incompleteness theorem is Gödel's completeness theorem, stating that:

If $A(\tau) \models \psi$, then $A(\tau) \vdash \psi$, or if A models a formula, it can prove the formula. It should be noted that this is a different kind of completeness to the meaning of completeness in the incompleteness theorems.

A proof of this theorem can be found in [25], pages 65-69.

There are an Infinite Number of Primes

Theorem: There are not a finite number of primes.

Proof: Assume that there are n primes from $p_1, p_2, ...p_n$. Construct a number $p_{con} = p_1 \times p_2 \times ... \times p_n + 1$. If p_{new} is a prime dividing p_{con}, then as p_{new} is not a member of $p_1, p_2, ...p_n$, we must have $n+1$ primes. This is a contradiction, and so for any number n it is not the number of primes. If p_{new} is not a prime dividing p_{con}, then there is no prime number dividing p_{con}; and so it is prime, which contradicts the assumption of there being finitely many primes. Therefore, there must be an infinite number of primes.

The Fundamental Theorem of Arithmetic

All natural numbers which are not 1 are either a prime number or the unique product of prime numbers. A proof of this theorem can be found in [2].

Consistency

Consistency is an important and generally assumed property in every system of maths or logic since without consistency we could prove anything; and while this might be gratifying, might not be very useful. We will always assume that PA is consistent. We will discover when examining the second incompleteness theorem that we cannot prove this internally and we are forced to assume it.

Definition: A model of a theory τ is **consistent** if and only if for all formula $\psi \in \tau$ we do not have $A(\tau) \vdash \psi$ and $A(\tau) \vdash \neg\psi$.

Definition: A model of a theory τ is **ω-consistent** if and only if for a general variable x there is no formula $\psi(x)$ such that $\psi(n)$ is provable $\forall n \in \mathbb{N}$, but $\forall x, \psi(x)$ is disprovable in

the model of the theory. For example, x could take the value of a negative number in an extension of PA; there are formulas which are provable for the naturals but disprovable in general for the integers, and so models of theories strong enough to define the integers are not ω-consistent [20].

Gödel Numbers

One of the important steps in proving the incompleteness theorem is finding a more systematic way of expressing logical formulae. We do this by assigning every symbol in our language a unique natural number; a Gödel number. This will allow us to easily express algorithms for testing the meaning of formula and making formal statements about the nature of these statements.

The exact assignments are unimportant, but we should note that we can create arbitrarily large numbers of variables, since for a variable θ_i, with assignment j, we can simply assign the next variable $j+1$. For example, assign 1 to $\vee$, 20 to θ_1 and 21 to θ_2. (An example of a full list of these assignments can be found at [18]).

We should also emphasize that we are going to choose a Gödel assignment which enables us to find Gödel numbers for every one of the axioms of PA. While it is not obvious that we can do this from our axiomatization, this is because we have chosen to increase the readability of the axioms over the rigour of exclusively using symbols. These axioms can be expressed symbolically, and there are many examples of this in the literature including many of the referenced books.

Let P be the sequence of primes, $P = 2, 3, 5, ...$ From a given formula made of primitive symbols ψ, put the first object in P to the power of the number assigned to the first symbol in ψ, put the second to the power of the number assigned to the second symbol in ψ, etc. The product of the powers of primes is the Gödel number for that formula. The number is unique, and the formula is retrievable due to the fundamental theorem of arithmetic.

For example, using the assignments from earlier, the formula $\theta_1 \vee \theta_2$ has a Gödel number of $2^{20} * 3^1 * 5^{21}$.

Let $Gn(\psi)$ be a function taking ψ to its Gödel number. Gn^{-1} is the inverse, such that for the Gödel number of a formula X, $Gn^{-1}(X)$ is the formula.

Gödel Numbers of Natural Numbers

There is a Gödel number for every natural number.

Proof: Every natural number can be obtained by applying the successor function to 1 a specific number of times. Therefore, every natural number n can be represented by $S(S(...S(1)...))$. This is a sequence of symbols; these symbols all have a Gödel assignment. As a result, n has a Gödel number encoding. Therefore, there is a Gödel number for every natural number.

The Provability Relation

The proof of a formula has a Gödel number. We know this because every axiom has a Gödel number, every formula that can be deduced from axioms has a Gödel number, and the deduction rules also have Gödel numbers. Since any proof contains only a sequence of these statements, we know that every proof has a Gödel number. This is important to us because we want to construct relations about which Gödel numbers prove which other Gödel numbers.

Definition: Let $Proof(X, Y)$ be a relation between X and Y, describing when Y is the Gödel number of a proof of the Gödel number of X. Alternatively: $Proof(X, Y)$ if and only if $Gn^{-1}(Y) \rightarrow Gn^{-1}(X)$ and the axioms of PA are a model for Y. $Gn(Proof(X, Y))$ exists, since this relation can be reduced to primitive symbols that we have Gödel assignments for. Since we can do this, we can also construct the Gödel numbers of formulas which say things about their proofs. This will become essential for proving the first incompleteness theorem.

Effective Methods

One of the principal advantages of Gödel numbers are that they make it easier to define effective methods for various useful transformations and deductions. This will help us prove that we can always determine what is a proof and what is not.

Definition: A **method** is a collection of rules which we follow to obtain an answer.

Definition: A method is **effective** for a class of problems if the method contains a finite number of instructions, and always stops after a finite number of steps for a problem within that class.

Although these definitions seem like they would be useful in complexity, the standard terminology in the literature changes for the different subject, and we will change with it.

An Effective Method for Generalisation

The generalisation deduction rule starts with a formula ψ with a free variable y which is provable in $A(\tau)$ (where y can take any value in our universe). From this, we obtain that; for all x, $\psi(x)$ is also provable. More formally: $A(\tau) \vdash \psi(y) \implies A(\tau) \vdash \forall x, \psi(x)$. ($\implies$ means deduction).

We want to find an effective method which takes the Gödel number of a formula where this rule can be applied to the Gödel number of the conclusion of the rule. We can always detect if what is in the premise of the deduction rule is present in the formula encoded by the Gödel number; if it is, we can simply replace the assignment value of y with the assignment value of x. Since we know ψ, we can add in $\forall x$ onto the two primes at the start of the encoding of ψ, shifting every Gödel assignment after $\forall x$ onward in the sequence of primes by two. A good example of a process similar to this can be seen next.

This satisfies the definition for being an effective method.

An Effective Method for Specification

While not strictly a primitive deduction rule, another elementary result that we can use to show the advantages of Gödel numbers is specification. The mathematical definition of this is: for $\psi(x)$, a formula with a Gödel number and a free variable x, for $n \in \mathbb{N}$ we can find $Gn(\psi(n))$. We will show that we have an effective method for this process as well.

Proof: As we saw earlier, n has a Gödel number, as does x. If we factorize $Gn(\psi(x))$ into its prime factors, and then, wherever x appears as an assignment value of a prime, we substitute that assignment value with the first assignment value in n, where n is represented as we saw earlier. We can then 'shift' as many future assignment values as necessary to make room for n.

For example, let $A[S]$ be the assigned value of the symbol S, and the same for every other symbol we have an assignment for.

Given a part of the factorization of a Gödel number of which x is a part,

$$p_i^{A[y_t]} \times p_{i+1}^{A[x]} \times ... \times p_{j-1}^{A[y_{u-1}]} \times p_j^{A[y_u]}$$

we can perform a substitution $x = 2$, obtaining:

$$p_i^{A[y_t]} \times p_{i+1}^{A[S]} \times p_{i+2}^{A[(]} \times p_{i+3}^{A[1]} \times p_{i+4}^{A[)]} \times ... \times p_{j+2}^{A[y_{u-1}]} \times p_{j+3}^{A[y_u]}$$

There are an infinite number of primes so we can always do this as (in the style of Hilbert's Hotel), there will always be enough primes to shift previous assignment numbers too. Since this algorithm always works, and algorithms are just another way of defining functions, we have proven that there will always exist a function f such that $f(Gn(\psi(x))) = Gn(\psi(n))$.

An Effective Method for Modus Ponens

We want to show that we have an effective method for applying the *modus ponens* deduction rule to the Gödel number of a formula, so that we can obtain the Gödel number of the formula that is deduced by the rule.

Proof: If we have $Gn(\psi, \psi \to \varphi)$ (we consider ψ and φ to be formulas in their own right here; or even lists of formulas connected by other deduction rules), and if we know ψ to be true, then we can deduce φ easily, since we can extract its Gödel number; and therefore obtain $Gn(\psi, \psi \to \varphi, \varphi)$.

An Effective Method for Determining the Provability Relation

It is necessary to be able to prove that that one of $Proof(X, Y)$ or $\neg Proof(X, Y)$ is always true. To do this, we need to show that there is an effective method which can always determine this. This proof is paraphrased from [6], page 118.

The axioms of PA have Gödel numbers, and so we can always determine whether or not a formula is an axiom. We do this by comparing the Gödel number of the formula with that of all the axioms; if it is the same, (or can be shown to be the same by some substitution of variables) then the formula is an axiom. Therefore we have an effective method for checking whether the formula is an axiom.

This is known as a system of axioms being recursively axiomatizable and provides two main reasons why there is the constraint that the system of mathematics must be at

least as strong as arithmetic for incompleteness to apply. Systems less strong then arithmetic cannot check whether their formula are axioms using only methods in that system; so all of the following would break down. Formal logic is an example of a system like this and is complete [12]. Theories which are not recursively axiomatizable cannot check whether every formula is an axiom at all, so without that constraint, we would also fail at this step.

From these two facts, we can define our method. The input must be the Gödel number X of a formula and the Gödel number Y of a proof of that formula; a sequence of the Gödel numbers of axioms and relations. This is a recursive method.

Check that all of the formulas and relations in $Gn^{-1}(X)$ and $Gn^{-1}(Y)$ are well-formed (we can do this effectively since this is simply checking a formula against a list of rules), and that $Gn^{-1}(X)$ is the final part of the proof. If this is not the case, $Proof(X, Y)$ fails.

Check that every formula in Y is either:

1. An axiom
2. A conclusion of two previous formulas which occurred earlier in the sequence as a result of *modus ponens* or generalisation; we have already shown that we have an effective method for these deductions.

If both of these conditions are true, then $Proof(X, Y)$ is true. Otherwise, it is false. Following these steps is an effective method for $Proof(X, Y)$.

The Gödel Sentence

The Gödel sentence is the key point in Gödel's original formulation of the theory. Although this is not the strongest formulation, the Gödel sentence is also used in the second incompleteness theorem and so it is worth discussing. It is important to recognise that neither the Gödel sentence nor the Rosser sentence have any free variables; their truth of falseness does not depend on the assignment of variables, only whether or not they are modelled by their theory.

Let τ_{PA} be a ω-consistent 1st order extension of PA. Since τ_{PA} is ω-consistent, any well-formed formula which is provable in PA is provable in τ_{PA}.

This is an important step in our proof since it enables us to generalise the incompleteness theorem for systems beyond PA. We know that we cannot take additional axioms

which prevent Gödel sentences from forming, and we shall soon see the importance of this.

Definition: A theory τ is **complete** if and only if for every formula $\psi \in \tau$, $A(\tau) \vdash \psi$ or $A(\tau) \vdash \neg\psi$. Or in English, a theory is complete if every formula is either provable or disprovable. A theory is **incomplete** if it is not complete.

Let the natural number G_s be the Gödel number of a (self-referential) formula such that: PA is a model for $Gn^{-1}(G_s)$ if and only if $\forall Y, \neg Proof(G_s, Y)$. This means that G_s is valid if and only if there exists no Y which is the Gödel number of a proof of $Gn^{-1}(G_s)$. This is the Gödel sentence, the specific statement which Gödel used to prove the original formulation of the first incompleteness theorem. Since $Gn^{-1}(G_s)$ and $\neg Gn^{-1}(G_s)$ are both formulas with no free variables that can be expressed in the language of PA, we assume (towards a contradiction) that PA is models one of them.

The following distinction will make the importance of this statement more apparent (we omit some cases here since they give the same results):

1. Assume that $\exists Y : Proof(G_s, Y)$. Therefore, PA does not model $Gn^{-1}(G_s)$, as a direct result of the definition of G_s. As a result, there exist statements which are provable in PA but are not modelled by PA. This violates our assumption of consistency, and so we accept that $Gn^{-1}(G_s)$ is not provable.

2. Assume that PA models $\neg Gn^{-1}(G_s)$. As a result, and because we have an effective method for the $Proof$ relation, we know $\neg(\forall Y, \neg Proof(G_s, Y))$. This is equivalent to saying that $\exists Y, Proof(G_s, Y)$. However, since PA is a model for $\neg Gn^{-1}(G_s)$, it is logically valid. Therefore, there must exist a proof due to Gödels completeness theorem. This contradicts ω-consistency.

3. PA does not model $Gn^{-1}(G_s)$ or $\neg Gn^{-1}(G_s)$. This contradicts our assumption, and entails incompleteness. We say in this case that $Gn^{-1}(G_s)$ is independent, as are all formulae with this same property.

So we obtain the following choice: PA is inconsistent, it is ω-inconsistent, or it is incomplete. We know that ω-consistency is a stronger condition then consistency, and so we can see that inconsistency is a larger defect then ω-inconsistency. We choose to have an ω-inconsistent theory. This means that Gödels original formulation of the first incompleteness theorem was: every ω-consistent, recursively axiomatizable extension of PA is incomplete. However, we will be able to improve this.

We should also note ω-inconsistency is not a very important constraint on our theory, and many theories are accepted as being ω-inconsistent, and are still widely used and accepted. For example, we know that every model which defines the rational numbers is ω-inconsistent, since the condition for this application refers in one place to variables which can take any value in τ_{PA}, and in another to variables operating only over the natural numbers. We have theorems in everyday maths which are provable only when we restrict ourselves to the natural numbers and disprovable if the theory is extended over numbers which are not naturals.

Rosser's Trick

My use of Rosser's Trick is adapted from [18].

We have almost proven the first incompleteness theorem; we have proven the same theorem that Gödel proved. However, Rosser's trick (proven in 1936 according to [9]) gives us a way to show that we can exchange ω-inconsistency for the more impactful condition of inconsistency in the result we just obtained. We could achieve the next result without proving the 1931 theorem which Gödel proved, but it is both historically and mathematically interesting to see both.

To show that we can do this, we will no longer be assuming that τ_{PA} is ω-consistent, but we will still assume that τ_{PA} is consistent.

Rosser's Provability Relation

We need to define a new relation, $Proof^*(X, Y)$. This relation is similar to the previous relation but with an additional constraint that the Y must be the least possible Gödel number that proves X, under the usual meaning of less than. We say that a proof $Gn^{-1}(Y)$ is shorter than a proof $Gn^{-1}(Z)$ if $Y < Z$. More formally: $Proof^*(X, Y)$ holds if and only if $Proof(X, Y)$ holds and $\nexists Z < Y : Proof(X, Z)$.

This new relation will be necessary for constructing the Rosser sentence and finishing this proof.

It is important to note that due to the uniqueness of Gödel numbers the only time two Gödel numbers are the same length are when the formulas they represent are identical. Therefore, any two formulas which say different things must have different Gödel numbers.

An Effective Method for Determining Rosser's Provability Relation

For similar reasons as before, we want to prove that we have an effective method for determining for any X, Y, whether $Proof^*(X, Y)$ is true or not.

Proof: We have already seen that we have an effective method for determining $Proof(X, Y)$ relation. Therefore, if we can find an effective method for determining which proof is the shortest, we will have an effective method for $Proof^*(X, Y)$.

Y is a Gödel number, and it has a length; the size of the number. We can test if $Proof(X, W)$ holds $\forall W \in \mathbb{N}, W < Y$. Essentially, we would check if every number less then Y is the Gödel number of a proof of X. If there is such a W, then we know that $\neg Proof^*(X, Y)$. So we can see that we have an effective method for determining $Proof^*(X, Y)$.

We have used Gödel numbers throughout this chapter, but rarely before has there been a more obvious example to point out their usefulness, since they give us an elegant way to find out the shortest proof of a theorem. If we were still operating in symbolic logic, we would struggle to find a way to compute the length of a proof.

The Rosser Sentence

We can then construct the Gödel number of a new formula R_s, such that R_s, is the Gödel number of a formula with the property: $\forall X, Y : Proof^*(R_s, X) \rightarrow (Proof^*(\neg R_s), Y) \wedge (Y < X))$. As before, since $Gn^{-1}(R_s)$ and $\neg Gn^{-1}(R_s)$ are both formulas that can be expressed in the language of PA, we must assume that PA models one of them.

A rough interpretation of the Rosser sentence in plain language is: If there exists the shortest proof of the formula R_s encodes, then we must be able to find a shorter proof of the negation of the formula R_s encodes. We can then finish the proof by considering the two scenarios.

Assume that R_s is provable. Since Gödel numbers are unique, there must be a shortest proof, and so $Proof^*(R_s, X)$ holds. Therefore, $Proof^*(\neg R_s, Y)$ and $Y < X$. This contradicts consistency since both R_s and $\neg R_s$ would be proved using the most general variables possible in τ_{PA} and every provable theorem must be modelled by τ_{PA}. The $Y < X$ does not affect this case.

Assume that $\neg R_s$ is provable. Therefore, there must be a shortest proof, and so $\exists Y : Proof^*(\neg R_s, Y)$. Since τ_{PA} is consistent, we know that $\nexists X < Y : Proof^*(R_s, X)$,

or: there does not exist a X less then Y which proves R_s. From this, we know that $Proof^*(R_s, X) \rightarrow (X < Y)$. This contradicts the definition of R_s. Therefore, if $\neg R_s$ is provable, τ_{PA} is not consistent.

Therefore, neither R_s or $\neg R_s$ is provable in τ_{PA}. Therefore, τ_{PA} is incomplete. Since τ_{PA} is a consistent extension of PA, (and we will never work in an inconsistent theory) we have now proven the first incompleteness theorem: every consistent, recursively axiomatisable extension of PA is incomplete.

The Second Incompleteness Theorem

The second incompleteness theorem is arguably the more important of the two. It follows using the result from the first incompleteness theorem, and now it will become slightly more obvious why we went through Gödels original proof since we will use his proof relation and the Gödel sentence rather then Rosser's improvements.

The Second Incompleteness Theorem: The consistency of Peano arithmetic cannot be proved inside Peano arithmetic.

It is important to note that the key point here is the idea of 'internal' proofs of consistency. The consistency of PA can be proven from the axioms of Zermelo–Fraenkel set theory with the axiom of choice (ZFC); but it can be shown that ZFC cannot prove the consistency of ZFC either.

Expressing Inconsistent Formula

The best way to express inconsistency is to portray a statement which the model of our theory will always output to false, and claim that it is true. If there is no case where we can do this, then our theory is consistent. If a theory is inconsistent, it is inconsistent everywhere, and so one inconsistent statement can represent all inconsistent statements.

The most obvious inconsistent statement we can pick is for some $n \in \mathbb{N} : n = S(n)$: or, a natural number is equal to the one after it. This statement has a Gödel number, and so do all other well formed formulas in our language; so every formula expressing PA's inconsistency has a Gödel number.

Using the Gödel Sentence

If a proof of PA's consistency exists, it is equivalent to disproving all formulae which express PA's inconsistency, such as $n \in \mathbb{N} : n = S(n)$. The first incompleteness theorem tells us that the Gödel sentence cannot be proven or disproven inside PA. We can (theoretically) enumerate the proof of the first theorem inside PA, in a similar way to our discussion of effective methods for provability.

If the Gödel sentence is true, then PA is inconsistent by the first incompleteness theorem. So if PA is consistent, PA can disprove the Gödel sentence. But if the Gödel sentence is false, (its negation is modelled by PA) then it is ω-inconsistent. Since the only numbers inside PA are natural numbers, this is equivalent to consistency; and so this is a contradiction. Therefore, PA cannot prove its own consistency.

Tarski's Undefinability Theorem

The undefinability theorem is much less well known than the incompleteness theorems, although it is sometimes said to have even more philosophical impact. Since it is much less well known, it is used more correctly then the incompleteness theorems, which is why I chose to prove the incompleteness theorems instead. It is also more ambiguous than the incompleteness theorems as to whether it is even mathematics, which means it barely fits inside the aims of this paper. It is strongly related to the incompleteness theorems, as they deal with the same subject of constraints on maths and logic.

Tarski's Undefinability Theorem: The definition of a model in a formal system which is strong enough to contain arithmetic cannot be defined within that system. A proof of this theorem can be found in [27], pages 152-278. A less formal interpretation of the theorem is: no system which contains arithmetic can define what truth means in that system. This is extremely disconcerting; the list of constraints on mathematics at least as strong as arithmetic has grown again.

It seems likely that this theorem would have a larger philosophical impact on the wider philosophical community if it was more well known then the incompleteness theorems since while provability might not be massively important to philosophers, definitions of truth certainly are. While its restrictions are great enough that it would not be able to resolve the debate on truth and knowledge for either side, it could certainly assist one or the other through arguments by analogy.

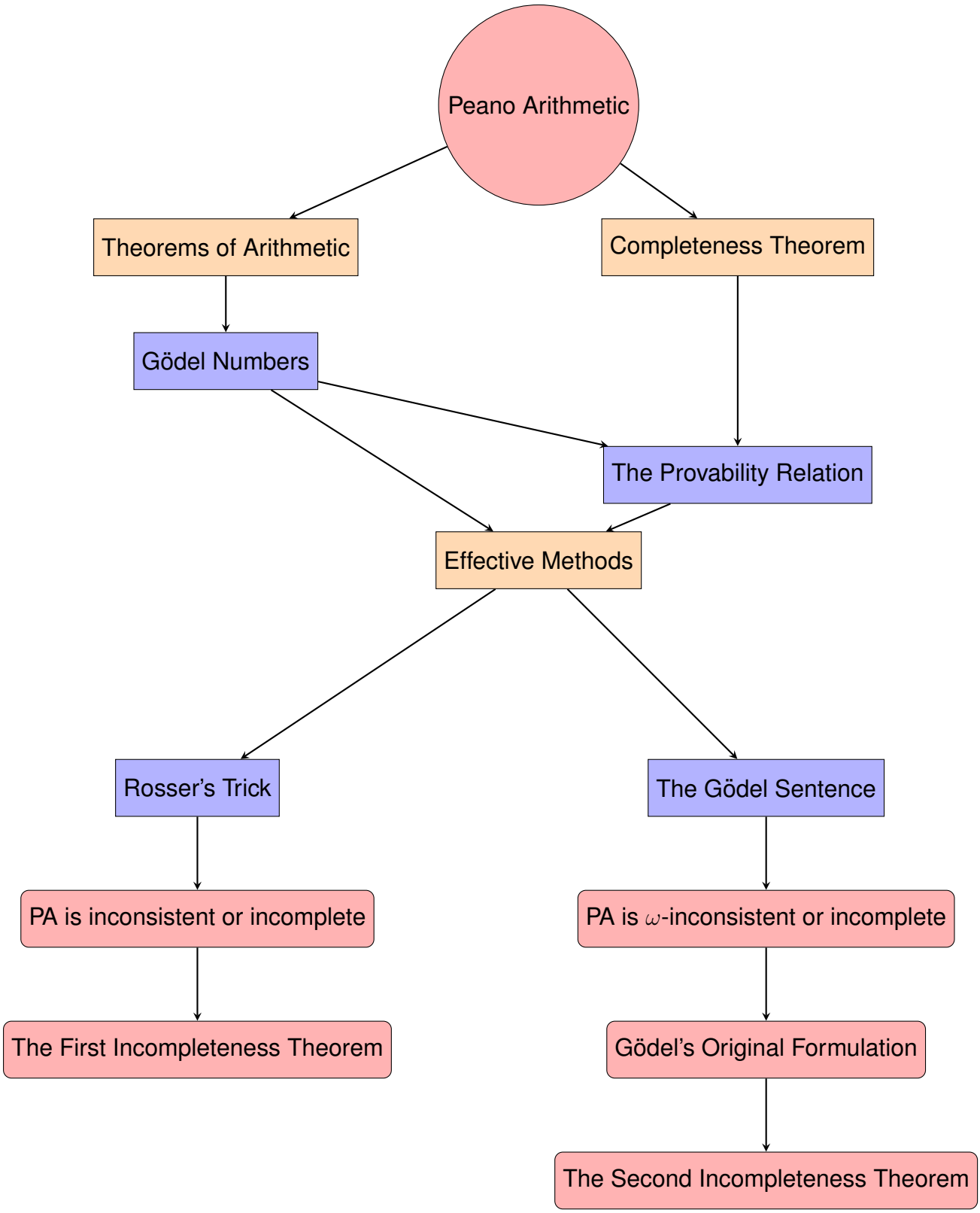

Figure 2: A Flow Chart of the Proof of the Incompleteness Theorems

Examples of Formulas Which are Independent of Their Models

We have seen in detail two formulas which cannot be proven or disproven inside their systems, the Gödel sentence and the statement of PA's consistency. We say that these formulae are independent of their systems. However, the fact that we cannot prove them would probably not bother the majority of mathematicians when doing normal mathematics. However, there are also formulae which have this property and are more impactful, and we use the axiom of choice and the continuum hypothesis as examples which impact the development of set and number theory respectively.

Both of these formulas are intuitively true. Like the triangle inequality, which says that the sum of the length of any two sides of a triangle is greater then the length of the third side, these formulas seem to be true according to our natural, intuitive understanding of mathematics. It is worth noting that human intuition does fail sometimes when it comes to mathematics (see the Monte Hall problem for an example), but it is usually accurate.

These formulae also have strong philosophical implications, and the approach to these theorems often depends on the philosophical view that the mathematician holds. If a mathematician investigating the axiom of choice was a platonist, they may claim that because it is intuitive, it is obvious that we can adopt it as an axiom of set theory, whereas formalists would only accept it if they were interested in the system which would be created by adopting it.

The Axiom of Choice

This section, and especially the diagram came from information and inspiration provided by my supervisor, Daniel Robertz.

The **axiom of choice** is a formula stating that: for every set containing some non-empty sets, there is a function which can select one element from each of them. This can include an uncountable number of sets which are themselves uncountable, and so accepting this axiom has significant consequences. However, it is equivalent to several other formulae, such as Zorn's Lemma and the well-ordering principle, which are less intuitive.

We can express this formula more formally in the following way: given a collection of

non-empty sets Λ, we can index each using points i on $]0, 1[$ and construct the set of these index's I. Consider a function f with the domain of the whole of I and the property $\forall i \in I, f(i) \in \Lambda_i$. The axiom of choice states that such an f exists for any selection of Λ.

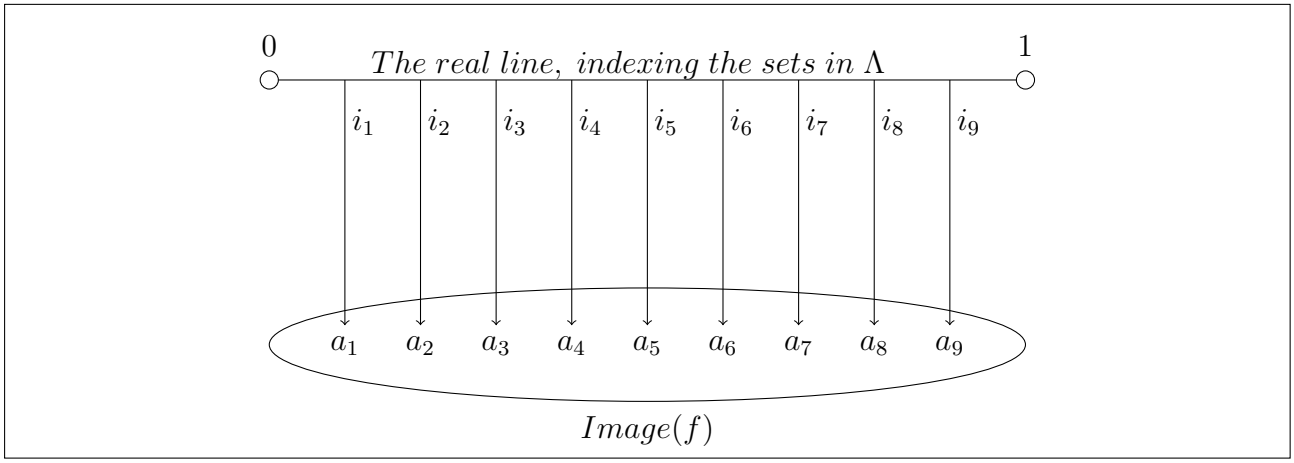

Figure 3: A Diagram of the Function Which Must Always Exist if the Axiom of Choice is True

The Continuum Hypothesis

The continuum hypothesis was proposed by Georg Cantor, the mathematician who contributed most to understanding different infinities. It also was the first of Hilbert's 23 problems [29].

Before understanding the continuum hypothesis, it is useful to recap the different orders of infinity. It can be shown that bijective functions can be created between natural numbers, the integers and the rational numbers. We often say that these sets of numbers are countably infinite, or equivalently that they have cardinal number 0; symbolised by $\aleph_0$.

It can be shown that there is no bijective function between any set in $\aleph_0$ and any set in $\aleph_1$ (such as the real and complex numbers). The continuum hypothesis is a formula saying that **there is no cardinality strictly between $\aleph_0$ and $\aleph_1$**, or: that there is no set of numbers Γ for which neither of $f : \mathbb{Q} \to \Gamma$ and $g : \Gamma \to \mathbb{R}$ is surjective, for any functions f and g. (Here $\to$ is used in the sense of a function, instead of meaning 'implies').

For our purposes, the continuum hypothesis is not the of interest for the implications it could have on set or number theory, but instead because in 1963 Paul Cohen proved that it could not be proved or disproved inside Zermelo–Fraenkel set theory. (A proof

of this theorem, the original paper, can be found in [3]). It was also shown later that the continuum hypothesis could also neither be proved or disproved inside of Zermelo–Fraenkel set theory with the axiom of choice [24]. The following is a diagram of the set that must exist iff the continuum hypothesis is false.

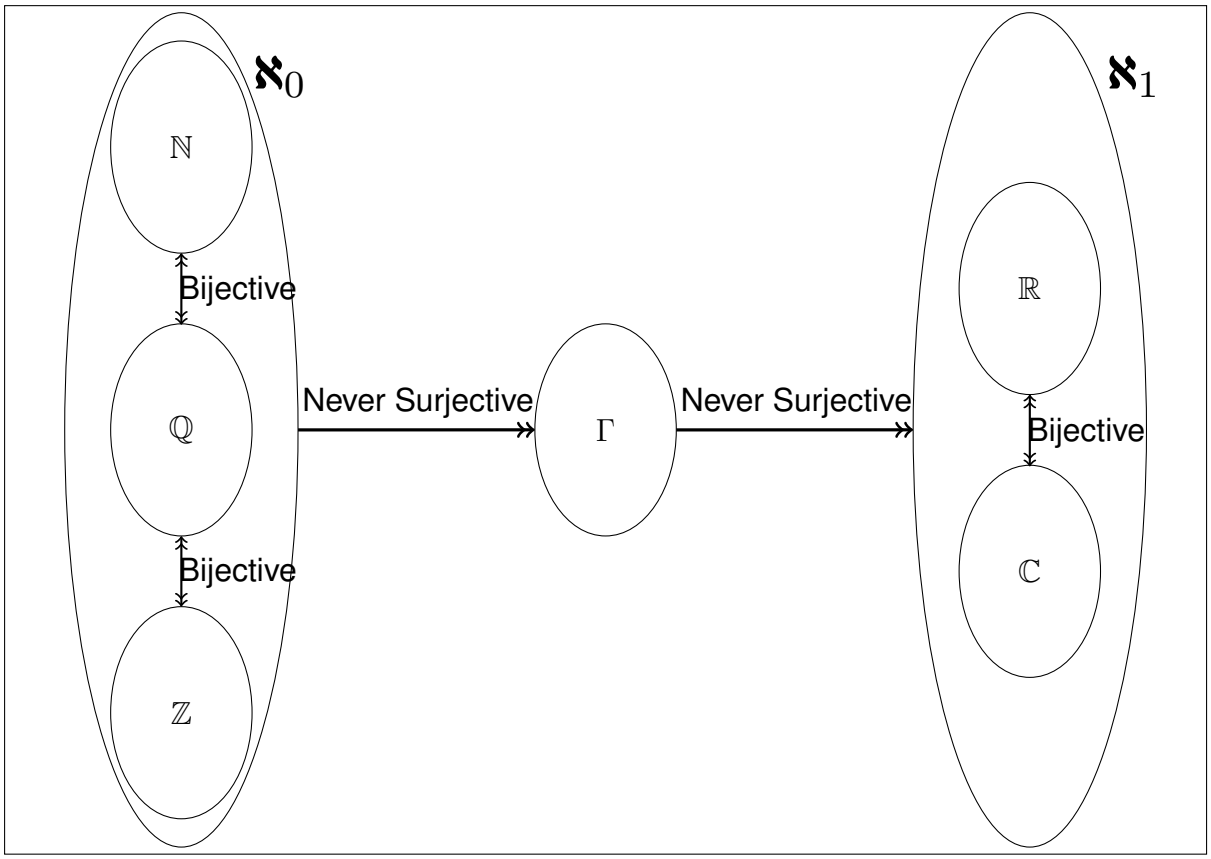

Figure 4: A Diagram Illustrating the Set Which Exists iff the Continuum Hypothesis is Set to False

Because the continuum hypothesis makes a statement about the non-existence of a set, a proof of the existence of such a set would be sufficient to prove the falseness of the hypothesis. This would contradict Paul Cohen's theorem. This has led many to believe that the continuum hypothesis is intuitively true, since we already know we would never be able to find a counterexample, although we cannot prove that such a counterexample does not exist.

Philosophical Implications

The acceptance of the axiom of choice was once controversial, but it is now used widely. However, unlike the continuum hypothesis, we can find specific examples (inside Zermelo–Fraenkel set theory) of the function we assert must always exist by accepting the axiom. Because of this, the axiom of choice seems more intuitive than the

continuum hypothesis, and certainly less eldritch.

There are different opinions on the importance of this revelation in the communities of mathematical philosophers. Formalists treat the continuum hypothesis as just another axiom; just as we excluded non-standard numbers from the natural set using the axiom schema of induction earlier, we can adopt the continuum hypothesis to exclude Γ from our system of mathematics.

Platonists would adopt the continuum hypothesis for a different reason; they believe it is objectively true, and that the $\aleph_0$ and $\aleph_1$ classes explain the world so well that considering the existence of a set (which is not constructable) strictly between them is not part of objective mathematical truth. We should note that intuitionists accept the continuum hypothesis out of hand; they would not recognise it as a legitimate question, once the fact that Γ could not be constructed was established.

Both the axiom of choice and the continuum hypothesis are accepted as 'good' axioms by most of the mathematical community whom they concern. However, there may exist an undiscovered axiom whose adoption is truly controversial, and which creates a schism inside a section of mathematics. It is possible to argue that some such schisms have already taken place in different areas, such as the difference between Bayesian and frequentist statistical methods.

Complexity

Complexity is a part of the study of the constraints faced when we or when computers carry out systematic processes with no user input or intuition, as well as how user-answered questions affect complexity. We will need to discuss the origins of computing before we can investigate these constraints. This chapter will also include some discussion of the apparent difference in difficulty between solving a problem and verifying its solution.

Turing Machines

The concept of a Turing machine was created by Alan Turing and became extremely useful when he was asked to assist in the Bletchly Park codebreaking operation during the second world war. This idea, along with the funding and assistance the circumstances gave him, allowed him to build what is often considered to be the first computer. The main other competitors for the title are Babbage's Difference Engine (a kind of calculator created in industrial era Britain) and the Antikythera Mechanism (A mechanism believed to be an orrery, constructed in ancient Greece).

A Turing machine is best defined by analogy, as follows (Paraphrased from [6], Page 35): Consider a tape strip which stretches to infinity in either direction, divided into a countably infinite number of discrete parts along with the tape. Each of the parts on the tape contains exactly one symbol. A Turing machine has an infinite list of internal states and is considered to be 'above' the strip of tape so that it can move along it in specific ways, and can read the symbols on their strips, as well as change them.

Now, we can describe the way a Turing machine operates: it starts in a specific internal state, and then reads the symbol it is positioned over. The internal state will have a specific set of instructions for what to do if receives this symbol; it could move the Turing machine to a different state, it could write a new symbol on the tape, or it could move the machine some number of places to the left or right, or some combination of the three, in any order.

These Turing machines give us a practical interpretation for what a function is, and allows us to treat piecewise, explicit functions the same as more mathematical ones which difficult otherwise. We can also see why this concept inspired computers since the internal states of the Turing machine represents the program of a computer, and the tape represents a computers storage. However, modern computers are more sophisticated, due to Random Access Memory, multiple cores, and a variety of other

innovations.

The following diagram is designed to help explain one interpretation of the mechanics of a Turing machine. The grey rectangle is capable of reading and writing the symbols (The author used the Hebrew alphabet since it is used rarely elsewhere in this paper) and the black line through it represents an axle capable of moving the rectangle left or right according to the symbols it reads.

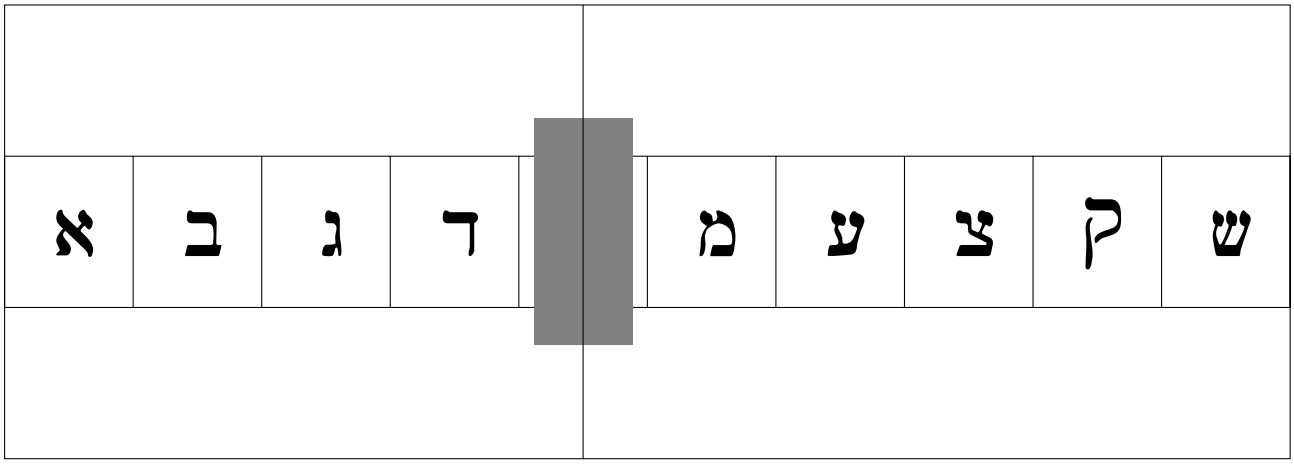

Figure 5: A Diagram Illustrating a Theoretical Turing Machine

Definition: A Turing machine is **Turing complete** or equivalently called a **universal Turing machine** if and only if it can recreate the process of any other Turing machine.

This definition makes it easy to talk about problems that no Turing machine can solve, as we can equate a problem no Turing machine can solve with a universal Turing machine not being able to solve the problem.

The Halting Problem

Definition: A Turing machine is said to halt if it reaches an output in a finite number of steps. If it does not do this, it would go on forever. We want to know if we can determine which Turing machines do halt.

The halting problem originally asked: **is there a general method of determining whether a given Turing machine will halt, rather than go on forever?** This was answered in the negative, there is no such method. I have provided a rough proof,

paraphrased from [6] for a more general audience.

Let H be a function of i and x where i is an index of a Turing machine in the set of all Turing machines and x is a specific input. Let H output 1 if the Turing machine represented by i halts for input x and 0 if it does not. Assume, towards a contradiction, that this function is well defined and possible.

We can then construct a Turing machine T which has input i and which halts if and only if i does not halt. One of the possible options for i is T itself, and so we can consider the result of H(T, T) (i.e. does T run with input T halt?). If T halts, T will not halt, and if T does not halt, T will halt. Informally, this makes H not well defined, and so we know we cannot construct a H that does this. Therefore, we know that we cannot construct a general function or algorithm which determines whether or not an arbitrary Turing machine will halt for a given input.

We can see that this proof follows similar lines to the incompleteness theorems, but it takes place in an entirely different style of mathematics. It is useful to see that the kind of proof used in the incompleteness theorems is not isolated inside addition, but can take place in widely differing systems of mathematics and logic.

Time and Space Complexity Classes

Now that we have the basis behind computers, we can examine the constraints when we try to use these machines to solve problems. I use the word problem to mean a question which we want to find an output for, given a certain input. These problems we want to solve include addition, ordering a list, finding the prime factors of a given number, determining if a formula is satisfiable, or predicting whether the price of a financial asset will increase or decrease.

We are interested in two constraints when we are talking about complexity: the time that the algorithm will take to run, which is usually assumed to be proportional to the number of the most difficult tasks that the algorithm performs, and the amount of computer storage the computer would take to run, which is analogous to how much space on the tape strip it would use.

The complexity of a program is best represented by a function: $F(n) = time$ or $G(n) = space$, where n is the length of the input (often considered as the length in binary, but the functions are independent of the units). F and G can be many kinds

of functions, and it is useful for us to split them into classes since this will tell us the behaviour of the constraints for large values of n. We determine the elements of these classes by the component which grows the fastest. For example, if $F(n) = log(n)$ then we would say that the problem represented by F is a part of the logarithmic complexity class. Any problem solvable in logarithmic time is also solvable in polynomial and exponential time, and so the class of problems solvable in logarithmic time is a subset of the classes of problems solved in polynomial or exponential time.

For theoretical purposes, we usually assume that F and G represent the constraint in the worst-case scenario, the hardest problem for the algorithm to solve. However, in practical applications, it is often of interest to find and optimize the complexity for the most frequent kind of problem instead of the worst-case one, or some other more complicated measure of value.

It is a common expression to say that a problem 'is part of the polynomial, exponential, etc. complexity class'. Often, what we really what we mean by this is that the most efficient Turing machine that has been found which can be used to solve that problem is of that complexity class. However, this is not always the case since there are proofs that certain problems have lower bounds for how quickly they can be computed, such as the problem of sorting an unstructured list [28].

The most interesting complexity class from our perspective is the polynomial time class, where the fastest growing term is of the form an^x where a and x are constants. This is because problems which take more than polynomial time to solve are considered computably unfeasible in most cases. This is not strictly a disadvantage; for example, encryption relies on the fact that its ciphers cannot be broken in polynomial time. It is still a problem which should be solved, however, both from a mathematical and an ethical perspective, since other problems such as in logistics also face this constraint.

Deterministic and Non-Deterministic Turing Machines

The kind of Turing machine we have defined up until this point have been deterministic Turing machines, which means that at any step they have exactly one option that they can follow. However, it is also useful to examine Turing machines which do not follow this rule, known as non-deterministic Turing machines. These have multiple instructions to follow at each given step, but they will always choose the 'luckiest possible option', meaning that it will always choose the shortest path to the correct answer.

Non-deterministic Turing machines are a purely theoretical existence, unlike deterministic Turing machines. However, they are useful in that they allow us to define the complexity class for a new kind of problem. If a deterministic Turing machine and a non-deterministic Turing machine with the same program are given the same input, then the deterministic Turing machine is solving the problem whereas the non-deterministic Turing machine is verifying a solution to the problem, since it already 'knows' the answer. We want to prove that this is mathematically true.

Formally (assuming we are only referring to polynomial time problems), the previous statement can be interpreted as: **we can use a Turing machine to verify a problem in polynomial time if and only if the problem can be decided by a non-deterministic Turing machine in polynomial time.** The proof of this theorem is rephrased from [22].

Proof: For the forward direction, we will assume that T is a problem which a deterministic Turing machine can verify in polynomial time. The non-deterministic Turing machine can correctly 'guess' the path to this solution, and the verifier can verify this solution (and its path) in polynomial time.

For the backwards direction, we will assume that T is a problem which a non-deterministic Turing machine can solve in polynomial time and that for any problem the non-deterministic Turing machine can solve, we know the path it will take (the 'lucky' choices it will make). Then we can construct a verifier which is deterministic, and yet relies on the path of the non-deterministic Turing machine to describe the steps the verifier should take. This leads the verifier to the correct answer. Since it only takes the steps the non-deterministic Turing machine does, it must be able to do this in polynomial time.

Reducing The Satisfiability Problem to the Clique Problem

This example was inspired by [22]. Some knowledge of graph theory may be required for this section, but understanding this example is not essential for understanding the rest of the discussion of complexity. This is an example of how we can reduce different problems to each other, and that there are some problems for which the output of two different Turing machines will always be the same; and how this can help us.

A clique is an object in a directionless graph, a collection of points on the graph in which every point is connected by an edge to every other point in the clique. The

clique problem is to determine whether a graph contains a clique of size k.

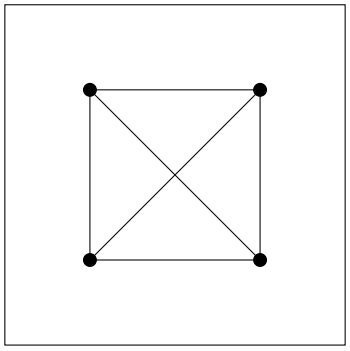

Figure 6: A Clique of Size 4

As before, we will assume that every logical formula can be represented in conjunctive normal form. We want to show that ψ, a formula with k clauses, is satisfiable iff the constructed graph G has a k-clique. Construct G as follows: for each clause in ψ, we assign a number of nodes equal to the size of the clause. Let every possible edge in G (assuming G is simple) be present except edges between nodes which come from the same clause and edges between nodes which cannot both be set to true by the model, such as x_1 and $\neg x_1$.

Suppose that ψ is satisfied by model A. Select one node in each specific clause generated subgraph which is set to true by A. If more then one of the variables which are transformed into this subgraph is set to true, we choose one arbitrarily. These selected nodes form a clique of size k, since we chose one from every clause and they cannot be contradictions if all are set to true in a satisfying model. This is the forward direction.

We also need to prove the other direction, in which we start with the same graph G and attempt to obtain the same result about ψ. Assume that the graph G contains a clique of size k, and reverse the construction so that we construct the formula ψ (it is easy to see that this construction is unique and reversible). All of the cliques' nodes are generated by variables in different clauses since no edge connects between nodes that come from variables in the same clause. We can choose an assignment in which the variable represented by any node in the clique is set to true, since there are edges between variables which directly contradict. Then since there is one such variable in each clause set to true, each clause is true; and so the formula is true, and is satisfied by the constructed assignment.

Therefore, every satisfiability problem can be reduced to a specific clique problem and vice versa. As a result, they must be in the same complexity class, and if we have an

algorithm for solving one in a certain time it solves the other in a certain time. (We also assume that we can construct G or ψ in this time class, and this is fairly obvious).

Turing Oracles

A Turing oracle is a concept which can be added to a Turing machine. It is essentially an internal state of the Turing machine, which unlike before, uses the input to ask a question and then use the output to carry on the Turing machine in the usual way. One of the practical interpretations is that the oracle asks the user a question, such as ”is it cloudy?”, and the oracle will output true or false depending on what the user says.

This also means that we have an easy way of describing a nested Turing machine since oracles can ask questions about the output of other Turing machines. This helps gives us an easy way to describe problems which can be reduced to each other. For example, a correct Turing machine for the satisfiability problem could be an oracle asking a question of whether there is a clique of a specific size in a specific graph.

P $\stackrel{?}{=}$ NP

We have discussed the class of problems that we can prove are solvable in polynomial time by a deterministic Turing machine, P. We can also use the idea we discussed earlier, non-deterministic Turing machines, to define a new complexity class; the set of problems which a non-deterministic Turing machine can solve in polynomial time, which we call NP. As we have shown, this is equivalent to saying that we can verify a solution in polynomial time. It should be clear from these definitions alone that P $\subseteq$ NP since solving a problem is sufficient to verify the solution.

The million dollar question is: is P=NP? We obtained one direction of this equality for free, so it only remains to prove that P contains every element of NP. However, this seems like an extremely daunting prospect since there are many problems in NP, unless we can find a trick to show it efficiently. There is such a trick involving oracles, but even so, the problem has not yet been decided.

NP-Completeness

We want to be able to make statements about the whole of NP, and this was historically accomplished through the use of a property of some elements of NP, known as

NP-complete problems. We should note that complete here does not mean the same thing as in the incompleteness theorems or Gödel's completeness theorem. It is unfortunate to have to use duplicate terms in this way, but these are used universally and I will follow the conventions of the mathematical community as far as possible.

An NP-complete problem is a problem where every other problem in NP can be reduced to it; so a correct Turing machine for a problem in NP is simply an oracle asking the outcome of a specific NP-complete problem. An NP-complete problem is at least as difficult as every problem in NP. This means that if we can prove that any NP-complete problem is solvable in polynomial time, P=NP.

Computers and the Resolution Calculus

The first problem that was shown to be NP-complete was the satisfiability problem. This is known as the Cook-Levin theorem since it was discovered by both Cook and Levin within a similar timeframe. A proof of this can be found at [5]. This is the problem which we attempted to solve using the resolution calculus. However the resolution calculus, like all other known theorem-proving procedures, takes place in exponential time [1]. Again, I will emphasize that we do not know whether this problem is in P; mathematicians have simply not been able to develop a method showing that it is.

However, we can show that the satisfiability problem is verifiable in polynomial time, by using the previous proof about polynomial time verifiers and non-deterministic Turing machines. Assume that a formula ψ has at least one model. A non-deterministic Turing machine can examine every variable in the formula, and choose an assignment for each leading to a model A such that $A(\psi)$ is true. If no such lucky guess exists, then ψ is not satisfiable. The number of steps involved must be a polynomial in terms of the input length since it only requires one step for each variable, and the other parts of the input ($\vee, \wedge$, etc.) do not affect the number of steps. Therefore, by the theorem we proved about deterministic and non-deterministic Turing machines, the satisfiability problem can be verified in polynomial time. As a result, we know that it is a member of NP.

Conclusions

We have two distinct conclusions to make; one summarising the mathematical process of this work, and the other the philosophical observations made. The following is an approximate summary of what we covered.

Mathematics

We have examined many of the constraints mathematicians face when doing maths, both immutable constraints and those which just increase difficulty. We started by examining a way mathematicians could automatically prove simple theorems, although in a language where we cannot quantify variables. Then, we proved the first incompleteness theorem, showing that even the arithmetic we all learnt as children has theorems which cannot be proven.

From here, we moved on to partially proving the second incompleteness theorem. We showed that any statement describing the consistency of Peano Arithmetic cannot be proved. We omitted first order language independent formula in favour of the axiom of choice and the continuum hypothesis.

The last section of this paper is complexity, the study of the limitations of systematic algorithms. This not only has some interesting theorems such as the halting problem similar to those we had already seen in a different system but also a practical constraint on the time and storage space a program requires. The apparent difference in difficulty between verifying the solution to a problem and solving it is also interesting, which we discussed briefly.

Philosophy

Neither Formalists nor Platonists have had their founding principles significantly affected as a result as a result of the results and theorems we have seen, although individuals have been forced to adapt and make their stances on various issues clear.

I would argue that formalists have been more affected by the results about incompleteness and the closely associated results of the independence of the continuum hypothesis, the axiom of choice and Tarski's undefineability theorem. This is mainly due to Hilbert's project being thwarted by the incompleteness theorems and the unusual results of the other theorems which appear to hint at a deeper meaning to mathematics.

Platonists are better at dealing with these new ideas. I cannot find evidence that Plato ever thought about the completeness of mathematics, and he did not make a philosophy based on it. Additionally, if truth in mathematics is more then just exploring axiom systems, then the constraints we have seen become almost expected.

Complexity has no clearly defined philosophical links to either Platonism or formalism, although since it is a new discipline these links may be developed in time. The question of whether or not computers can truly understand things is interesting since if they can then they can understand proofs, however, this question is outside the scope of this paper.

Culmination

After so many pages and words, we have finished this work. I was more prolix than I had hoped, but I believe I achieved all of my aims. We have been able to explore many of the constraints of mathematics and explained them rigorously for those who are not mathematicians. We have also explored some background philosophy for mathematicians, and it is hoped that this will facilitate and inspire dialogue between these groups.

Acknowledgements

I would like to thank my supervisor Daniel Robertz, for all his esteemed help, guidance, advice and proofreading. I would also like to thank my friends, who filled the time concurrent to writing this with laughter.

A Glossary of Notation

This glossary is not an exhaustive list of all symbols used in this work; but it covers those which are important, used frequently, or not widely known. Symbols which are used universally in the literature are labelled 'Standard notation', otherwise 'Defined in this work' is used, although I may have been inspired to use the notation from other works. It is sorted by order of appearance, as far as it is possible.

Symbol	Meaning	Origin
P	Polynomial time class	Standard notation
NP	Non-deterministic polynomial time class	Standard notation
P $\stackrel{?}{=}$NP	A famous problem	Standard notation
$\theta_1, \theta_2, ...$	Propositional variables	Defined in this work
$\wedge$	Logical AND	Standard notation
$\forall$	For all; variable quantifier	Standard notation
$\exists$	There exists; variable quantifier	Standard notation
ψ, φ	Formulas	Defined in this work
$x_1, x_2, ...$	Universal variables in any system	Defined in this work
$\vee$	Logical OR	Standard notation
$\rightarrow$	Logical IMPLIES	Standard notation
i, j	Counters	Standard notation
$\neg$	Logical NEGATION	Standard notation
S	Successor function	Defined in this work
$\in$	An element of	Standard notation
τ	A general theory	Defined in this work
A	A general model	Defined in this work
$\models$	Models	Standard notation
$\vdash$	Entails/Proves	Standard notation
Φ	A theory containing specific formulae	Defined in this work
$a, b, ...$	Non-standard numbers	Defined in this work
$\mathbb{N}$	The natural numbers	Standard notation
n, m	Natural numbers	Standard notation
L	A propositional variable or its negation	Defined in this work
$\square$	The empty clause	Defined in this work
Res	The resolution operator	Defined in this work
K	A set of clauses	Defined in this work
$C_1, C_2, ...$	Clauses	Defined in this work
$\setminus$	Set minus	Standard notation

R	A resolvent	Defined in this work
Gn, Gn^{-1}	Operators involving Gödel numbers	Defined in this work
$Proof$	Gödel's proof relation	Defined in this work
X, Y, Z	Gödel numbers	Defined in this work
G_s	The Gödel sentence	Defined in this work
$Proof^*$	Rosser's proof relation	Defined in this work
$\aleph_0$	Indicates countability	Standard notation
$\aleph_1$	Indicates uncountability	Standard notation
$\mathbb{Z}$	The integers	Standard notation
$\mathbb{Q}$	The rational numbers	Standard notation
$\mathbb{R}$	The real numbers	Standard notation
$\mathbb{C}$	The complex numbers	Standard notation
T	A Turing machine	Defined in this work

Table 1: Table of notation

References

[1] Matthias Baaz and Alexander Leitsch. *Complexity of Resolution Proofs and Function Introduction*. *Annals of Pure and Applied Logic*, 57:181–215, 1992.

[2] R. P. Burn. *A Pathway Into Number Theory*. Cambridge University Press, 1982.

[3] Paul J. Cohen. *The Independence of the Contiuum Hypothesis*. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 50 (6):105–110, 1963.

[4] S Marc Cohen. *Second-Order Logic*. [Online]. Available from: https://faculty.washington.edu/smcohen/120/SecondOrder.pdf. [Accessed 31/10/19].

[5] Stephen A. Cook. *The Complexity of Theorem-Proving Procedures*. *Proceedings of the Third Annual Association for Computing Machinery Symposium*, pages 151–158, 1971.

[6] S Barry Cooper. *Computability Theory*. Chapman and Hall/CRC, 2017.

[7] Dilts. *Gödels Incompleteness Theorems*. [Online]. Available from: https://infinityplusonemath.wordpress.com/2017/08/04/godels-incompleteness-theorems/), 2017. [Accessed 07/12/19].

[8] Behnam Esfahbod. *Euler Diagram for P, NP, NP-Complete, and NP-hard sets*. [Online]. Available from: https://en.wikipedia.org/wiki/P_versus_NP_problem#/media/File:P_np_np-complete_np-hard.svg). [Accessed 21/12/19].

[9] Torkel Franzén. *Gödel's Theorem: an Incomplete Guide to its Use and Abuse*. AK Peters/CRC Press, 2005.

[10] R. L. (Reuben Louis) Goodstein. *Essays in the Philosophy of Mathematics.* Leicester U.P, 1965.

[11] Russell A Gordon. *Real Analysis: A first course*. Addison Wesley, 2001.

[12] LH Hackstaff. *Systems of Formal Logic*. Springer, 1966.

[13] Leon Horsten. *Philosophy of Mathematics*. [Online]. Available from: https://plato.stanford.edu/archives/spr2019/entries/philosophy-mathematics/. [Accessed 12/2/20].

[14] The Clay Math Institute. *Millenium Problems*. [Online]. Available from: https://www.claymath.org/millennium-problems). [Accessed 29/01/20].

[15] Irving Kaplansky. *David Hilbert*. [Online]. Available from: https://www.britannica.com/biography/David-Hilbert). [Accessed 29/01/20].

[16] Stephan Kor̈ner. *The Philosophy of Mathematics : an Introductory Essay.* Hutchinson & CO (Publishers) LTD, 1960.

[17] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[18] Panu Raatikainen. *Gödel's Incompleteness Theorems*. [Online]. Available from: https://plato.stanford.edu/entries/goedel-incompleteness/. [Accessed 20/9/19].

[19] Margaret Rouse. *First-Order Logic*. [Online]. Available from: https://whatis.techtarget.com/definition/first-order-logic. [Accessed 31/10/19].

[20] Robert Seely. *Using the Gödel Sentence to Prove the Incompleteness Theorem*. [Online]. Available from: http://www.math.mcgill.ca/rags/JAC/124/theorems. [Accessed 2/11/19].

[21] Saeed Salehi & Payam Seraji. *Gödel–Rosser's Incompleteness Theorem, Generalized and Optimized for Definable Theories*. [Online]. Available from: https://doi.org/10.1093/logcom/exw025), 2016. [Accessed 14/11/19].

[22] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[23] D.M. Smirnov. *Encyclopedia of Mathematics*. [Online]. Available from: https://www.encyclopediaofmath.org/index.php/Model_(in_logic). [Accessed 12/11/19].

[24] Raymond M Smullyan. *Set Theory and the Continuum Problem*. Clarendon Press Oxford University Press, 1996.

[25] Shashi Mohan Srivastava. *A Course on Mathematical Logic*. Springer Science & Business Media, 2013.

[26] Plato (Translated by Benjamin Jowett). *Meno*. [Online]. Available from: https://www.gutenberg.org/files/1643/1643-h/1643-h.htm), 2008. [Accessed 12/12/19].

[27] Alfred Tarski (Translated by J.H. Woodger). *Logic, Semantics, Metamathematics*. Hackett Publishing Company, 1983.

[28] School of Computer Science University College Cork and Information Technology. *Problem Complexity*. [Online]. Available from: http://www.cs.ucc.ie/ dgb/courses/toc/handout28.pdf). [Accessed 05/02/20].

[29] Eric W. Weisstein. *Hilbert's Problems*. [Online]. Available from: http://mathworld.wolfram.com/HilbertsProblems.html). [Accessed 29/01/20].

[30] Bertrand Russell & Alfred North Whitehead. *Principia Mathematica*. Cambridge University Press, 1910.

[31] Eliezer Yudkowsky. *Standard and Nonstandard Numbers*. [Online]. Available from: https://www.lesswrong.com/posts/i7oNcHR3ZSnEAM29X/standard-and-nonstandard-numbers. [Accessed 22/10/19].