Set Theory, Type Theory and the future of Proof Verification Software

James Palmer

3rd Year Mathematics and Philosophy undergradate

University of Warwick

James.G.Palmer@warwick.ac.uk

July - August 2020

Contents

1 Preface and Introduction to the Xena Project

This essay was written to introduce readers to foundational ideas of set theory and type theory, which are not as widely known as they should be. This essay will also introduce readers to the power of Lean, discuss the questions we should ask about what Lean implies for the future of mathematics as well as what impacts it could have in the ongoing philosophy of mathematics.

The end of the essay will raise another discussion as well: the pragmatic compromises needed to be made between man and machine in order for the progression of mathematics, and the enhancement of mathematical research via automated theorem proving, ensuring its large potential is fulfilled.

This essay will be a worthwhile read for any undergraduate student studying either mathematics or philosophy and hopefully an enthusiastic sixth-form/ high school student will be able to take away something worthwhile as well.

The *Lean project* was originally launched in 2013, at Microsoft Research Redmond by Leonardo de Moura. Paired with *mathlib*, both allow an open source library of automated proofs to whomever wants to harness it. It allows us to formally verify proofs usually made in mathematical terms by using methods from both logic and computer science. Formal verification allows us to check that these proofs are true by computer, which is important in a mathematical age where proofs of the Poincare conjecture and Fermat's last theorem have been the length of novellas. Human error within these proofs is more and more likely to be overlooked by the fellow humans verifying the proof by eye, hence it is clear to see that such formal verification is destined to be the future of mathematics. Famously, Andrew Wiles' initial proof of Fermat's last theorem had an error in it that needed to be corrected, something Wiles would've spotted himself were he using a proof verification software. Computer software, especially with big data, is becoming more and more integral to the way science is done and maths is surely going to follow suit.

Most mathematics, although rigorous, is explained through terms which imply the use of formal logical rules but does not explicitly express them. The extra effort thus has to be made when feeding them to proof-verification software due to the fact that we have to use a formal language to make these logical inferences explicit for the software to verify the proofs properly. Lean somewhat sidesteps this issue by introducing a 'tactics' mode, allowing us to express the mathematics somewhat closer to how we would write a mathematical proof in an exam for a human to then mark.

The Xena project was set up by Kevin Buzzard, with the goal of encouraging mathematicians to use proof verification software (Lean being the most used proof verification software at the moment). Its current goal is to formalise all of the Imperial College London (where it is based) mathematics course and then

feed the resulting database to an AI, seeing where the AI takes the maths from there. It is worth noting that having the mathematics required by a standard undergraduate mathematics course available to anyone by Lean will be a massive help in verifying proofs at higher levels. This essay is not intended to be a tutorial for how to use Lean, there are many better documents you can find in the wild which would do a better job at that which are mentioned in the further reading section. The majority of good resources for learning Lean can be located on the main website for The Xena Project. This essay has the main purpose of explaining a couple of the key foundational issues that still plague mathematics, due to the fact that more undergrads and prospective mathematics students should know these issues as they affect the very way we do mathematics. In places, we will also be showcasing the power of Lean as a method of formalising mathematics.

Due to the ongoing pandemic that will last throughout the summer months, Buzzard's previous plans which involved hosting a number of undergraduate students at Imperial College London to formalise mathematics found in most contemporary undergraduate mathematics courses around the world into Lean had to be cancelled. However, due to the digital nature of what these students had to do, it was easy not only to transition the summer projects to a digital platform, but also to allow even more students worldwide to take part in their own formalization projects. This led to this essay's inception. Here we will cover important ideas within set theory, type theory and intuitionistic maths that I think every mathematician and analytic philosopher should know about.

I would like to thank Kevin Buzzard for the opportunity to write this essay in association with the Xena project as well as his help with improving it, David Loeffler for reading through and offering invaluable critiques of the essay, as well as giving me my first taste of defending my arguments to an actual expert with 'Professor' in their title. Thank you to Luca Seemungal for introducing me to Lean in the first place and checking over drafts of this essay for me. Lauren Nicholson must also be thanked for her suggestions on how to make this essay more readable and for casting her watchful eye over my grammar. I would also like to give a special mention to everyone who has contributed to Lean and the Xena project, automated theorem proving is the future of mathematics and you are the trailblazers.

2 Introduction

Throughout history, humans have engaged in mathematical practices and this practice has become more and more advanced in its technique and methods. Mathematics has become a framework with which we make advancements in the sciences through use of measurement, make inferences based on the mathematical applications of statistics and probability in the social sciences and use to model markets in economics. All of these applications and more are derived from what we discover in mathematics.

Yet, there are very valid questions that precede our applications of mathematics, which is to say *where does mathematics come from? What makes up the bedrock of mathematical reasoning? What is a mathematical object? What is a proof?* These questions are important for the mathematician because it influences the way we then decide to practise mathematics (as we will see in how intuitionistic logic works). These questions are also important to the philosopher because of how important the discipline of mathematics is to questions in epistemology and the philosophy of language. An argument about how we come to gain knowledge or how language is made meaningful to us can live or die based on the application of it to mathematics. Throughout history, the most well-known thinkers from both mathematics and philosophy have offered their answers to these questions, with varying degrees of success and influence on our contemporary discussions.

Two of the positions that philosophers of mathematics take on the origins of mathematics are the ideas of logicism and intuitionism, which will be discussed at length here. Logicism, if we were to condense the ideas into a single slogan, states that all of mathematics is built from ideas found in classical logic, i.e. using ideas such as the truth values and derivability of sentences involving the logical laws that come from using the connectives $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$, defined (respectively) as 'and, or, not, implies, equivalence' to form the statements we would find in all branches of mathematics, where $P \wedge Q$ is true if and only if P and Q are true, $P \vee Q$ is true if and only if either of P or Q are true, $\neg P$ is true if and only if P is false, $P \rightarrow Q$ is true if and only if Q is false or both P and Q are true, $P \leftrightarrow Q$ is true if and only if P and Q are equivalent statements (i.e. P is true if and only if Q is true). Predicate logic expands on this. We then introduce variables such as x, y, z, the universal quantifier $\forall$ and the existential quantifier $\exists$. The universal quantifier effectively translates to "for all" and the existential qualifier effectively translates to "there is". Hence $\forall x P(x)$ is true if for every constant t, $P(t)$ is true and $\exists x P(x)$ is true if there is one constant term t such that $P(t)$ is true. For example, "all the natural numbers are even" is false, as 3 is odd, but "there is a natural number" is true due to 2 being a natural number (among other choices).

One of the key rules of classical logic that dictates a large amount of our mathematical practises is the law of excluded middle, written formally as $P \vee \neg P$.It

is a tautology which states, when harnessed in mathematical practise, that for every mathematical statement P, either P is true or $\neg P$ is true. This idea is key to performing a proof by contradiction in mathematics, where we assume the negation of a statement in order to derive a contradiction.

From the logicism movement, set theory (in ZFC form) and type theory (in Bertrand Russell's ramified type theory) were born in an attempt to capture this underlying structure by which all mathematics is done. These theories, as well as their many variations which won't be covered here, have produced results with varying degrees of success.

The intuitionists, with their main ideas also condensed into a single slogan, state that mathematics is not the study of object universal truths independent of humanity, it is instead the study of our own mental constructs that guide our perception of the world around us. They also believe that it is arrogant for us to assume that a mathematical statement always be proven or disproven. A mathematical statement is true to the intuitionists if a valid proof can be constructed for the statement. A key part of intuitionist practise is rejecting the law of excluded middle, due to the fact that the intuitionists believe it entails mathematical Platonism, a philosophical view which states that mathematical objects are real and exist in a world of the forms which we cannot perceive. This leads to many proofs which were previously doable via the use of proof by contradiction now being undoable.

Early intuitionistic logic managed to prove that every function whose domain and codomain are the real numbers is continuous. Intuitionistic logic also leads to intuitionistic type theory via the Curry-Howard correspondence, a big topic in both mathematics and computer science, so there is no doubt that it has positive effects on mathematical practice. Through the way proofs are constructed, intuitionistic logic allows us to read said proofs as algorithms which we could then go on to use as programs. This is the key central element that makes the Curry-Howard correspondence so powerful and what makes intuitionistic logic worth study.

But what is a proof? To the mathematician, a proof will be something much different to what it is to the logician. In mathematics, we mainly see proofs as being the method by which we show that statements are true or false, yet I would be willing to put money on the fact that if you asked the average mathematics student what a proof looked like, they would only be able to point to examples of proofs rather than give a satisfying abstract answer. It is only really in philosophy, with the study of logic, that we get a true understanding of what a proof actually is.

In mathematics, proofs are mainly are series of logical deductions written in prose, with steps being outlined as to how we go from our hypotheses to our conclusion. Typically, a mathematician will be building a theory. Hence entire

sub-proofs are stored in 'lemmas' before a theorem is proved by building around the results of said lemmas. A classic type of proof in maths is a proof by induction, where if we prove that a result being true for n means that it is then true for $n+1$, as long as we prove that the result holds for 1, we prove it holds for every natural number n. Another type more commonly used is proof by contradiction, where we make a statement, assume that the statement is false and show that this leads to contradiction.

In logic, proof theory and the formal study of what a proof is form a key part of the academic field. Natural deduction systems are studied, which show us the rules by which all proofs are logically valid. These are the rules that underpin mathematics, with any proof using the rules found in classical proof systems such as the Fitch proof system.

These proofs are much more technical than the prose found in mathematical proofs, yet they are essential in proof-verification software as computers do not (currently) reason through a paragraph of prose as we would. Hence there is an inherent obstacle here when it comes to getting mathematicians to use Lean, as mathematicians must learn to write proofs in much more detail. A solution is found in the 'tactics' mode in lean, where methods of going through multiple steps of a proof can be saved to a couple letters. 'The Natural Numbers Game' found on the Xena Project's website introduces you to tactics almost instantly. Later on when we encounter the proof that the real numbers are uncountable written in Lean, we will see lemmas and tactics saved to words allowing us to write the entire proof in one easy line of code.

It is worth noting now that when formalizing mathematics onto Lean, we are using a version of type theory known as *dependent* type theory as our formal basis of mathematics, which uses something known as a propositions-as-types correspondence using intuitionistic logic simultaneously as a functional programming language. Notably, we are not using classical logic at all at the basis. This may surprise some readers, due to set theory being more wildly popular and widely used among mathematicians. However, dependent type theory offers a huge practical advantage while programming due to the fact that it requires us to be more specific with types we are using, as we will see later. It is worth noting, however, that Lean allows for classical reasoning to be done when formalising mathematics, which has contributed a lot to its popularity over theorem formalisers like Coq.

Having characterised the main ideas that will be involved in this essay, it is time for us to move onto the foundational theory that almost every reader will be vaguely familiar with, namely set theory.

It is worth noting now, for those who haven't seen, that Appendix C contains a glossary which the reader is encouraged to use if they get lost.

3 Set Theory

In the British education system, students will be introduced to set theory by the visualisation of it through Venn diagrams, to the point where students are likely to believe that sets are nothing but Venn diagrams. This, as with most things taught to us in high school, turns out to not be the entire story. Even then, the set theory used in high school maths is mainly nothing but naïve set theory, without the necessary axioms in Zermelo-Frankel-Choice set theory which allow us to give a formalized definition to sets and avoid paradoxes.

The logicist movement as well as a desire to study infinity is what propelled set theory into existence. The goal of the logicist movement was to show that mathematics could be written by using the rules of logic. As set theory manages to use first-order logic in order to express its axioms and thus by extension a large amount of mathematics, this was seen as the goal for a lot of mathematicians with regards to finding a foundational theory of mathematics. Of course, set theory has evolved since this, as mathematics has evolved and other theories besides logicism have come to light, hence ideas that compete with set theory have been present since its inception.

A set, informally, is just a collection of objects. The objects themselves could be sets and in ZFC, every object is a set. A member of a set is known as an element and to express that an element x belongs to a set X, we write $x \in X$. A subset of a set A is a set B such that for every element $x \in B$, $x \in A$ as well. This is commonly denoted $A \subset B$. A function f is a mapping between two sets A and B, this is usually denoted as $f : A \to B$, with A referred to as domain and B referred to as a codomain. Hence for each element $a \in A$, we have an element $f(a) \in B$, where $f(a)$ is equal to some $b \in B$, where b is the element that a is mapped to by f.

For those unfamiliar with what an axiom is, in the dictionary an axiom is defined as a 'self-evident truth'. In maths, the term axiom is used more to mean a rule that does not need to be proven upon which we then base the rest of our theory. While we would love these axioms to be self-evident truths in order to feel like we are unearthing the secrets of the universe by which all natural and logical laws are dictated, we will see in practice that set theory and type theory both have their fair share of axioms that instead are put in place more for convenience rather than for presenting a self-evident truth. In fact, both Ernst Zermelo and Bertrand Russell both developed their axioms in response to various paradoxes and inconsistencies which Cantor, in his development of set theory, and Frege, in his development of logicism, had run into. The most famous of which was Russell's paradox, which states (if we put it in naive set-theoretical terms) that there exists a set $A = \{X | X \notin X\}$, i.e. A is the set of all sets X such that X is not a member of itself. Yet $A \in A$ if and only if $A \notin A$, which is clearly an unwanted equivalence.

The axioms are as follows, do note that an 'axiom schema' is just a collection of axioms:

Assume that A, B, u, v, x, y are sets. Then we have:

The Axiom of Extentionality: $\forall A \forall B((\forall x(x \in A \leftrightarrow x \in B)) \rightarrow A = B))$
(If two sets have the exact same elements in them, then they are the same set.)

Axiom of the Empty Set: $\exists B \forall x(x \notin B)$
(There exists a set with no elements)

Axiom of Pairing: $\forall u \forall v \exists B \forall x(x \in B \leftrightarrow (x = u \vee x = v))$
(For every pair of sets, there is a set that contains them both)

Axiom of the Power Set: $\forall u \exists B \forall x(x \in B \leftrightarrow x \subset u)$
(For every set u, there exists another set whose members are the subsets of u)

Axiom Schema of Comprehension: Let ϕ be any formula that does not contain B and it only has bound variables excepts for $x, t_1, t_2, ..., t_n$. Then we have the following Axiom:

$\forall t_1 \forall t_2 ... \forall t_n \forall A \exists B \forall x(x \in B \leftrightarrow (x \in A \wedge \phi(x)))$.

This axiom lets us avoid Russell's paradox as it means we have to be much more specific about how we build subsets. It is important to note that these x must be a member of the original set A. $\phi(x)$ is just a generalised notion of a first-order logic formula, effectively stating 'x must have predicate ϕ. An example of this would be 'x is even' or 'x is a square number'. So what the formula is saying is that there are subsets of a set which satisfy certain predicates.

Axiom of Union $\forall A \exists B \forall x(x \in B \leftrightarrow \exists y(x \in y \wedge y \in A))$
(i.e. given two sets, there is a set which contains all the members of both sets and nothing more.)

Axiom Schema of Replacement: $\forall A((\forall x \forall y \forall y'(x \in A \wedge \phi(x, y) \wedge \phi(x, y')) \rightarrow y = y')) \rightarrow \exists B \forall y(y \in B \leftrightarrow \exists x(x \in A \wedge \phi(x, y)))$
(i.e. the image of a function is a set)

Axiom of Infinity: There is an inductive set; that is,
$\exists A(\emptyset \in A \wedge \forall x(x \in A \rightarrow x^+ \in A)$
(i.e. there exists a set with an infinite amount of members. x^+ is defined as the successor of x, e.g. 1 is the successor of 0, 2 is the successor of 1, 3 is the successor of 2, etc)

Axiom of Choice: Let A be a set of pairwise disjoint non-empty sets (which could be infinite). Then there exists a set C containing exactly one member of

A; that is for each $B \in A$, $C \cap B$ has only one element.

These axioms became so popular due to the fact that we could now express almost all of mathematics through statements in first-order logic. First-order logic is an example of a formal language, which try to capture how humans reason in an abstract mathematical way. What we do is choose a model which may contain a domain, functions, relations and constants as well as a language with which to express the model, and then using sentences within the language, we can express theories by using abstract symbols.

It is a remarkable achievement of logic and mathematics that these axioms, which are consistent with each other, not only manage to build up through the technicalities of elementary number theory so well, but they even manage to capture the most complicated parts of mathematics today. Dynamical systems, mathematical analysis, hyperbolic geometry and topology are some of the many fields which ZFC manages to capture. In fact, ZFC set theory manages to express virtually everything that mathematics has to offer.

These axioms manage to cover a large amount of modern mathematics, however they come with weaknesses. It is commonplace for philosophers of set theory to complain about either one or more of these axioms, with complaints usually being about the power/lack of power of each axiom. By far the most controversial of all axioms is the axiom of choice, mainly for the central fact that the axiom of choice asserts that certain sets exist. However, unlike the other axioms, it does not give us any hint as to what these sets end up looking like. For more discussion of the axiom of choice, see Appendix A.

In addition to these axioms, we can also represent functions as sets! To represent these mappings as a set, we use something called 'ordered pairs', which are written $\langle x, y \rangle$. By using a set of ordered pairs $\{\langle a_1, b_1 \rangle \langle a_2, b_2 \rangle, \dots\}$, where $a_i \in A$ and $b_i \in B$ and $f(a_i) = b_i$, we can thus represent a function as a set. This is the simplest example of a part of mathematics being modelled in ZFC set theory, yet considering how functions are arguably as important as sets to forming the bedrock of mathematics, it may well be one of the most important.

The key strength of set theory and these axioms is that, for the most part, these axioms look intuitive. While some, comprehension especially, test the limit of 'self-evident truth', for the most part ZFC gives us sets and rules that we would want and expect a conception of the foundation of mathematics to have.

Of course, however, there are more drawbacks than just the axioms. As much mathematics as this manages to model, sadly there are still some parts of mathematics that set theory fails to cover. The most famous example of a hypothesis which is independent of the axioms of ZFC set theory is the continuum hypothesis. A hypothesis is independent of a collection of axioms if it is neither

provable nor disprovable from the axioms. Both the continuum hypothesis and the negation of the continuum hypothesis are consistent with ZFC set theory, yet neither of them are provable. This is an application of a very famous result in mathematical logic known as "Gödel's incompleteness theorems", where for certain collections of axioms which are consistent, there is another sentence which is consistent with these axioms, yet we cannot use our rules of natural deduction to craft a proof of this sentence from these axioms.

The continuum hypothesis states that there is no set whose cardinality is greater than the cardinality of the natural numbers (which we refer to as aleph-naught) and the less than cardinality of the real numbers (which we refer to as the continuum). Both these sets have an infinite number of elements, yet the set of the natural numbers is countably infinite, i.e. it is possible for us to list out its members $\{1, 2, 3, ...\}$, whereas the set of the real numbers is uncountably infinite, i.e. it is impossible for us to list its numbers out, as any list we make using finite generators will not contain every member of the set. The following is a showcase of the proof of the fact that the set of real numbers is uncountably infinite - done in Lean.

```
lemma not_countable_real : ¬ countable (set.univ : set ℝ) :=
by {rw [countable_iff, not_le, mk_univ_real], apply cantor}
```

Listing 1: A remarkably consise proof of the uncountability of the reals in Lean produced by Floris van Doorn. The 'rw' function rearranges the previously proved theorems to prove the statement above.

Due to the fact that the continuum hypothesis is independent in ZFC, it mean that a number of other assertions in other areas of mathematical study like analysis and point-set topology are also independent of ZFC due to their reliance on it. It seems that even when we do assume the law of excluded middle, $P \vee \neg P$, we still run into the issue that it is not always enough to tell us which of P or $\neg P$ is true, as for some theories such as ZFC set theory, both P and $\neg P$ are consistent with the theory and thus there is still a limit to its power. In some ways, mathematicians who wish to stick with classical logic over intuitionistic logic have to realise it's not a case of accepting or rejecting the law of excluded middle, it is rather picking your poison in the limitations of what the law of excluded middle can tell us.

4 Type Theory

Type theory has evolved arguably more than set theory has as a foundational idea during the last century. Set theory evolved from Cantor trying to study the infinite by harnessing mathematical techniques to becoming a foundational idea due to the ideas and axioms of Zermelo and Fraenkel. There are multiple different versions of set theory, such as descriptive set theory and fuzzy set theory, which have the aim of covering different parts of mathematics as well, yet ZFC set theory remains the most well-known of all approaches to set theory.

Meanwhile, type theory begins with the ideas of Bertrand Russell and Alonzo Church, two titans of mathematical logic. Through Russell's ramified type theory and Church's lambda calculus (an evolved version of which Lean utilises in its programming language), type theory was born. Meanwhile, due to Per Martin-Löf's work on intuitionistic type theory, modern type theory, usually referred to as intuitionistic, dependent or homotopy type theory (with each having their own variation on the theory), has an intuitionistic basis and is used much more frequently in computer science for functional programming than it is in mathematics. This change of foundational basis from classical to intuitionistic logic is why it has evolved much more substantially than set theory has.

What type theory has retained throughout is its specificity - our ability to distinguish between types of mathematical object that we are dealing with. This has a legion of advantages when it comes to formalising mathematics, which we will illustrate now in the following example.

Within set theory, we ascribe sets to natural numbers as a way of defining them, as every mathematical object is a set and thus natural numbers are no different. Hence, we have the following:

0 is represented by the empty set, $\emptyset$.
1 is represented by the set containing the empty set, $\{\emptyset\}$
2 is represented by the set containing both the set of the empty set and the empty set, $\{\{\emptyset\}, \emptyset\}$
3 is represented by the set containing the set containing both the set of the empty set and the empty set, the set containing the empty set and the empty set, $\{\{\{\emptyset\}, \emptyset\}, \{\emptyset\}, \emptyset\}$

A similar pattern follows for the rest of the natural numbers. What is interesting to note here is that there are implications that may then be made about the relationship between these numbers. For example, those of you who are aware of the definition of a topology may notice that the set representing the number 3 meets the definition of being a topology on the number 2. Topology is a field of study concerned with geometric properties that preserve under continuous deformations, so it seems strange that suddenly we have topologies on the natural numbers to some mathematicians. Thus due to this, we have

many different contexts where we could be referring to the mathematical object "3". Is it the natural number? The integer? The rational number? The real number? Is it even the topology on the number "2"?

The lack of specificity regarding an object is something that would be nothing less than a nightmare to deal with in a programming context, so in type theory, we can assign types to the mathematical objects we denote instead. This does mean that in pure mathematical terms, the number "2" belonging to the type 'natural number' and the type 2 belonging to the type 'real number' are two separate different mathematical objects. The following is a representation of some very bad Lean code, for we have defined constant 'a' to be of four different types.

```
constant a : nat
constant a : ℚ
constant a : ℝ
constant a : ℂ
```

In set theory, we could get away with this as if $a \in \mathbb{N}$, then a belongs to all of the other sets as well. Meanwhile in type theory, due to us having to specify the type that the constant belongs to, it is not equal to the same constant in a different type. Effectively, we have got rid of the idea of subsets in type theory as we knew them in set theory.

```
constant a : ℚ
```

Listing 2: A simple example of code that is actually usable in Lean

Hence, type theory is remarkably better than set theory for building programming languages on due to its specificity, thus it is also a much more helpful foundational mathematical theory to use when formalising a proof into proof verification software. In terms of how we write proofs in type theory, this will be covered in chapter 5 after we have met the Curry-Howard correspondence. The main difference from writing formal set theory proofs is the fact that set theory uses first-order logic as the formal language of its proofs, yet the Curry-Howard correspondence allows us to write proofs by using types themselves.

Another advantage for early type theory is how it really takes care of 'impredicative definitions', something Bertrand Russell was especially keen to tackle because of how Russell's paradox (named after him after his discover of it at the turn of the 20th Century), arose from the use of impredicative definitions in his view. An impredicative definition is a definition which defines itself in relation to a collection of objects. For example, the 'supremum' of a set is defined to be the least (smallest valued) upper bound (a number M such that for all $x \in X$, $x < M$) of the set, hence we define a number here from a collection of numbers. After defining individuals as being of type 0, classes (which are a collection of individuals) being of type 1, classes of classes being of type 2, we move on to the hierarchy of types which gives Russell's type theory the name

of 'Ramified Type Theory'.

Russell went on to introduce another hierarchy of types, dealing with impredicative definitions. He introduced a hierarchy of orders, so for any type n, we can have a type n class of order 0, a type n class of order 1, etc. A class is of order 0 if it is predicative, a class is of order 1 if the class is not predicative, but it can be defined in terms of predicative classes (e.g. a class whose members were the supremums of other classes), a class is of order 2 if it is not of order 0 or order 1 but can be defined in terms of order 1 classes, etc. So for example, a collection of individuals that could be defined in terms of order 1 classes would be considered a type 1 class of order 2.

Thus, something which appears epistemically problematic at first is dealt with in a neat way. Using Lean, we can express a similar hierarchy of types but not of orders and thus show the descendants of Russell's works implemented into the language. Using the `check` function in Lean, we can see where on the hierarchy of types any type is.

```
/- Note that for any n in the natural numbers, type n will
be of type n + 1 as type n is a collection of type n - 1
classes. -/

constant α : Type
constant β : Type 45

#check α -- Type
#check β -- Type 45
#check Type  -- Type 1
#check Type 73494 -- Type 73495
```

Russell's type theory did not come without its issues however, as is common place for any theory tackling foundational mathematics. The axioms were called into question when they did not seem like a self-evident truth, but rather they seemed more like something thrown into a theory to cover up paradoxes and deficiencies within it.

The axiom which caused the most issues for Russell's type theory specifically was the axiom of reducibility. It states that for each type of a class C, there is an impredicative class I with the same members as C. So seemingly, we begin to ignore the structure of orders and suddenly pretend that impredicative issues do not matter.

This axiom was born out of the issue that we have no confirmation that the set of the natural numbers has the same definition at each order of the hierarchy, which is an obvious issue due to the fact that having one set with two (or more) unidentical definitions is a massive problem. A natural number n is originally defined by Russell as a `Type 2` class; it is the class of classes which

contain exactly n individuals. The natural numbers are defined as the class of these classes. Hence from this, it is unclear how we would then define the natural numbers in an identical way at various orders.

Another early version of Type Theory was Alonzo Church's introduction of types through the lambda calculus. The lambda calculus works in a similar way to first order logic, however instead of using sentences featuring atomic sentences, constants, variables, quantifiers, connectives and parentheses, we have terms made up of atomic terms, applications and abstractions. The core idea to the lambda calculus is that functions are our objects of study, as it is a model of computation. This gives us a framework with which to study computable functions (i.e. functions where we have an effective process for deciding an output, given an input) in a subject called *computability theory*. Computability theory makes up the theoretical basis upon which all current computation is based and is worth its own essay, hence we will mention it only where necessary.

What Curry found was that by applying a theory of types to the lambda calculus, the programs written in the lambda calculus represented proofs made in intuitionistic logic. This became known as the Curry-Howard correspondence, as people realised that intuitionistic proofs and programs have a one-to-one correspondence.

The type theory applied to the lambda calculus for this is very similar to the dependent type theory covered in Chapter 5, hence we will go into it a lot more there. For now, however, do note the examples of ways we would express functions in the lambda calculus. The type theory introduced over these ends up having atomic types (which we would consider analogous to sets in set theory) and function types (which we would consider analogous to functions in set theory).

```
constants α β γ: Type -- atomic types
constant a : α -- expressing individuals as belonging to a
type
constant f: α → β -- f is a function belonging to function
type α → β
constant g: α → Type -- g is a function belonging to
function type α → Type
-- all of the above are atomic constants, hence they are
terms in λ-calculus
#check f a -- application, this allows us to get a term of
type β from a term of type α and a term of type f: α → β.
#check λ x : α, f x -- abstraction over a term, this allows
us to get a term of type f: α → β from a term of type α and
a term of type β.
#check λ (α : Type), g -- we can abstract over types
themselves, this has type 'Type'.
/- Examples of functions expressed in λ-calculus -/
```

```
constants m n : nat
constant (m,n): nat × nat
constant g: nat → nat
#reduce (λ x : nat, x) n -- returns n, this is the identity
function
#reduce (λ x : nat, m) n -- returns m, this is the constant
function
#reduce (λ x : nat, x + n) m -- returns m + n, the addition
function
```

5 Dependent Type Theory

Before leaping into the more technical aspects of dependent type theory, it is worth understanding where the intuitionists lie philosophically. As mentioned above, the intuitionists reject the law of the excluded middle. L.E.J Brouwer is known as the founder of this position within the philosophy of mathematics. The story has it that he was famously unimpressed with Gödel's incompleteness theorems because they had been published around 15 years after he had already denounced the law of excluded middle for its faults.

Brouwer stated that the law of excluded middle inherently assumes that the mathematical objects we refer to are real objects. To explain why he thought this, it is worth looking at an analogous example that Russell is famous for using in an argument regarding the philosophy of language. If I were to say "The King of France is bald", then is this statement true or false? Due to the fact that there is no King of France, the statement itself is meaningless, so how does a statement like $P \vee \neg P$ possibly make sense if P does not exist in the first place? Russell solved this issue in the philosophy of language with his theory of definite descriptions, which we will not divulge into here. It is worth noting that his theory does not solve the issue of using the law of excluded middle's use (or misuse...) in mathematics.

The issue here is that without knowing it, mathematicians are walking into mathematical Platonism. This is the belief that mathematical objects are real and mind-independent, existing on some higher realm of being than physical objects do. This originates from Plato's idea of the world of the forms, where every physical object has a corresponding form in the world of the forms.

Due to this inherent mathematical Platonism linked with using the law of excluded middle, Brouwer rejected it. Instead, intuitionism was born as a form of mathematical constructivism. Constructivism itself is the belief that a mathematical proposition is true if and only if there is a proof of it.

It is worth mentioning that when we think of the ontology of a proof and whether we should take a realist stance (the proof exists and is mind-independent) or an anti-realist stance (the proof only exists in our perception and is not independent of human observation), this question is bound to be controversial. On the other hand, the question of whether a program is mind-independent or not is bound to come to a much more agreeable conclusion. Programs themselves are just processes we get computers to enact as a routine, computers which have been built in the first place by humans. Thus, it makes sense to use intuitionistic logic when programming proofs, for the fact that when converting a proof into a program in order for it to be formalised, we have an anti-real proof represented by an anti-real program as well. To convert a mind-independent proof into a mind-dependent program would be a fallacious philosophical jump.

Intuitionistic type theory is where a lot of modern interest in type theory lies, due to the intersection between mathematics and computer science lying in it, especially with its uses in functional programming. As stated above, due to the specificity that comes from being able to ascribe a certain type to an object, instead of all objects being sets, makes it much more practical as a programming language, especially when it comes to formalising mathematics.

The origins of intuitionistic type theory could potentially be ascribed to Per Martin-Löf, with his 1989 paper titled 'Intuitionistic Type Theory'. Inspired by the Curry-Howard correspondence, he lays out a syntax, semantics and proof theory for intuitionistic type theory. Historically, it is worth noting that he developed this as both a typed functional programming language as well as a proof system due to the one-to-one correspondence between proofs and programs mentioned at the end of Chapter 4. A lot of these rules introduced here can be found in the dependent type theory used in Lean, including the rules regarding Π-types and Σ-types. We will explain $\Pi-$types and showcase its usefulness in Lean.

For set theory, we explained how we model functions using ordered pairs. If we then wanted to express a set of functions $f : A \to B$ then we would create a set of sets of ordered pairs, which may seem a bit cluttered, but it works. What we instead have in dependent type theory is the idea of f instead being a function type written as $f : \alpha \to \gamma$, similarly written as it is in set theory, mapping individuals in a type α to a type γ. So let us now define α to be an arbitrary type and define β to be a function type $\beta : \alpha \to Type$. For each a of type α, $\beta(a)$ denotes a different type, so depending on how many individuals a belong to a type α, that same number will be the amount of different types $\beta(a)$ we are working with.

We then use the type $\Pi x : \alpha, \beta x$ to denote the type of functions f with the property that for each individual a belonging to type α, $f(a)$ belongs to the type $\beta(a)$. Hence what we are interested in is the type of $f(a)$, which depends on the input a. For the fact that the value of β depends on the parameters set by the individuals belonging to type a and there is no fixed codomain, this is why dependent type theory is given its name, as $\Pi x : \alpha, \beta$ is also known as a Pi-type or a dependent function type. Effectively, this is a generalisation of a function type, as instead of a function type mapping individuals from one type to another type, a Π-type maps individuals from one type to many types instead.

This is only one example showcasing a dependent type however, sigma types $\Sigma x : \alpha, \beta x$ are another example and offer a generalisation of what are known as cartesian products $(\alpha \times \beta)$ and are another example of a dependent type, hence they are sometimes known as dependent products. Both Sigma-types and Pi-types are very important because they can be used to model the existential quantifier and the universal quantifier respectively, as was their purpose when introduced by Martin-Löf in intuitionistic type theory. In fact by using

Pi-types, we can express logical propositions as types, aiding how we express proofs through it and, by extension, how we express them in Lean. Propositions-as-types is a key feature of intuitionistic type theory, as this feature is directly influenced by the Curry-Howard correspondence.

```
constant α : Type
/- We want to fine a way to describe predicates of α, in the
same way we can use ∀xP(x) in first-order logic to describe
predicates in set theory. We use the following.-/
constant p: α → Prop -- Prop is the type representing
propositions, with our propositions-as-types magic being
showcased here
#check p x -- this is a proposition, so it makes sense for
it to be of type Prop! set theory and ramified type theory
do not allow for such fluidity as propositions-as-types does.
∀ x : α, p x -- assertion that p holds of x for every x,
this is an element of Type. The truth value is dependent on
input x, hence we use some dependent type theory machinery!
```

As a reminder, in natural deduction systems, systems in which the goal is to formally develop rules how to construct a valid proof, to introduce ∀ into a proof, we use the introduction rule below:

∀ Introduction - Given a proof of `p x`, in a context where x is arbitrary (i.e. choice of x doesn't matter as it's just some indiscriminate variable), we obtain a proof $\forall\ x : \alpha,\ p\ x$

If instead we have a statement including ∀ and want to get rid of it, we use the following rule.

∀ Elimination - Given a proof $\forall\ x : \alpha,\ p\ x$ and any term `t` : α, we obtain a proof `p t`

For Pi-types, let us consider their introduction/ elimination rules for proofs within the natural deduction system used in dependent type theory.

```
/- Intro - Given a term t of type β x, in a context where x :
 α is arbitrary (i.e. choice of x doesn't matter as it's just
some indiscriminate variable) we have (λ x : α, t) : Π x : α
, β x -/
/- Elim - Given a term s : Π x : α, β x and any term t : α,
we have s t : β t-/
```

Notice the similarities. From Curry-Howard correspondence, it is clear to see that Π-types are the program equivalent to how ∀ is used in proofs of propositions. Hence we use Π-types in our Propositons-as-types correspondence in order to program in ∀ into proofs.

```
/- Now we will demonstrate a program for a natural deduction
```

```
    law called 'modus ponens', which states that if we have a
    proof of P → Q and a proof of P, then we have a proof of Q-/
    constant Proof: Prop → Type
    #check implies -- equivalent of P → Q in first-order logic,
    function type of Prop → Prop → Prop (as it is a binary
    connective, it takes two propositions and returns a new one)
    Π p q : Prop, Proof (implies p q) → Proof p → Proof q
    /- This program states for every pair of hypotheses P and Q,
    if we have a Proof of P and a Proof of P → Q, we have a
    proof of Q. The equivalent of modus ponens!-/
```

In Lean we can also construct what are known as inductive types, where we inductively define types with use of the distinctions between type 0, type 1, ... , type n, ... by using a number of finite generators. We have seen this already in Chapter 4 when discussing the hierarchies and definitions found in Russell's Ramified Type Theory. We can use these to construct new types as well. For example, below is a method of constructing the natural numbers in Lean.

```
    inductive nat
| zero : nat
| succ (n : nat) : nat
```

Listing 3: succ is the successor function so succ n is equeal to n + 1. Note the similarity with the axiom of infinity.

A question worth dissecting is the question we started with, does the law of excluded middle really lead to mathematical realism? If we choose to accept it, then are we aligning ourselves with the view that mathematical objects exist independently from us? One could easily make a case for the fact that while "the present King of France is bald" certainly does imply the existence of a King of France, should we not be more careful with the ontology of mathematical objects? Which, from everything we can tell, do not share the same ontology as physical objects such as the present King of France would do, were they to exist? The fundamental question at play here is whether or not ideal objects which exist as theoretical objects in our mind alone all share the same ontology with each other, which is a philosophical can of worms about the nature of being that I will leave the reader to dissect.

6 The future of Lean and philosophy of mathematics

There are currently two schools of thought within the mathematical community regarding the future of automated theorem proving software. There are those who believe it is the future of mathematics: these academics usually have research interests in algebra where automated theorem proving has been adopted more. As well as those who believe it is a small trend that will die out, usually analysts who have not seen it used in their research circles as much. This has been the indication given to me by the conversations I and other people flying the proof verification software flag have had with various mathematics academics.

I personally believe that it will be used more and more in the future, yet it will only grow in popularity as quickly as we formalise the mathematics we currently know. This is happening, but this is not happening at a lightning speed. Particularly, topics in analysis are currently not as thoroughly developed as topics in algebra are. Perfectoid spaces and the continuum hypothesis have both been formalised into Lean, both requiring some very heavy algebraic machinery behind them. Meanwhile, it took a very long time to even prove that the derivative of $\sin(x)$ is $\cos(x)$ using Lean, something proved within the first year of an undergraduate course.

The fact that the formalisation of analysis is happening more slowly than the formalisation of algebra is why algebraists are more enthusiastic about proof verification software than analysts are, as academics in algebra will know of more colleagues that are currently adopting proof verification software than academics in analysis. Mathematicians being wholly pragmatic beasts (hence why classical logic and set theory are still predominantly used over intuitionistic logic and type theory, because mathematicians could not care less for foundational issues), I imagine that as proof verification software is developed further and further (potentially becoming more user-friendly on the way), more mathematicians will begin to use it. Kevin Buzzard of the Xena Project has said before that he does not believe there is someone on earth who fully understands the entire proof of Fermat's Last Theorem due to its length and use of very complex mathematical ideas, however theorem provers such as Coq and Lean will be able to spot any mistakes with ease.

While Lean specifically allows for the user to use classical reasoning over intuitionistic reasoning, due to the need to have a knowledge of dependent type theory in order to properly get to grips with the language, this will confront mathematicians to at least briefly think of the foundations of mathematics and the issues regarding the use of the Law of Excluded Middle. As proof-verification software becomes integral to more and more academics, it will almost certainly begin to be taught in undergraduate mathematics courses (in my personal opin-

ion, it definitely should be at the moment). Mathematicians who wish to embrace the future may have to make a compromise, sacrificing their time to learn enough dependent type theory and functional programming to be able to use Lean or another proof verification software. However, Lean is moving in a really exciting direction and in time, it could become a tool which accelerates the pace of academic research into the subject.

There are, of course, compromises that the machine may also have to make as well. Currently, mathematicians who are already deep into research will not have the time to use such software due to the commitment to having to learn about dependent type theory and the ways in which Lean works. This is a compromise which certainly seems to be being made, Kevin Buzzard has developed a number of games (the further reading section will tell you where to find these!) which allow you to learn how Lean works while concerning you with dependent type theory as little as possible. It is worth noting that Lean rose to prominence as the proof verification software of choice due to the fact that it allows for classical reasoning to be used, something which allows mathematicians wishing to use Lean without having to learn as much dependent type theory as they would otherwise. It is still fundamentally important to the system though.

While this essay has deeply covered Lean's involvement with foundational issues, it is worth noting that Lean's ability to help us in mathematics research reaches far beyond the questions it may make us reconsider about philosophy of mathematics. For example, Kevin Buzzard cares far less about foundational issues than I do, but his main motivation to start formalising mathematics into Lean was due to the fact that errors in mathematics may not just appear at the very foundational levels of maths, it could appear on higher levels as well.

To use an analogy that David Loeffler used when I discussed the matter with him, a skyscraper can fall over due to error made at the foundations of the construction of the skyscraper, however it is just as much of a disaster if an error at the 37th, or 86th floor of the skyscraper causes the higher levels to topple. Lean allows us to not only verify theories built off of the axioms with which we build all of our mathematics in foundational theories, but also the axioms we use to build many other parts of mathematics as well. Propositions we make whose proofs use lemmas built off of the axioms of group theory, analysis proofs built off of the completeness axiom, propositions anywhere using any sort of lemma or theorem in their proof, virtually all can theoretically be formalised and verified in Lean.

In my view, even allowing undergrad mathematicians the option to learn how to use proof-verification software in their courses would be a great way to future-proof mathematicians, as mathematicians with proof-verification software proficiency in their arsenal will be at a clear advantage in a future where proofs are going to get longer and more susceptible to error. Even in smaller proofs, there are a number of reasons why someone may make an error on a proof, but

there's no doubt that it happens.

By teaching the foundational mathematics ideas necessary to fully understand dependent type theory and Lean, mathematics undergraduates would then have to think more about which of logicism and intuitionism captures the ontological nature of mathematical objects. At the moment, perhaps your university maths society or your friend who does a Mathematics and Philosophy degree may inform you about the discrepancies found at the foundational levels of mathematics. It would be encouraging to see Lean emerge as a mainstay in undergraduate mathematics courses and thus the conversation regarding these foundational ideas becoming more prevalent. Students will no longer be indoctrinated into accepting the ideas of set theory non-thinkingly as they are now.

Of course, there are a number of philosophical implications that come from either adopting a classical or intuitionistic view of mathematics which will briefly be explained here. These concern the ontology of mathematics as well as various ideas about how we learn mathematics.

Of course, the plan is that once Lean has an undergraduate course worth of mathematics formalised into it then an AI will be fed the mathematics and we will see where that manages to take us. This is something of special interest in our current times, as AI is beginning to have a more prominent role in many industries. What implications does the philosophy of mathematics and computation have for AI?

For the fact that we have to inherently use constructivist/ computable mathematics in any computer software that is used, Lean also constrained by this, it is clear that any question proved incomputable by computability theory is going to thus be impossible for any computer to do. This not a mainstream topic of study, yet due to how pertinent it is to the foundations of computer science, as well as considerations with regards to how much time/ energy it takes to compute certain algorithms, anyone attempting to answer questions regarding the possibility of AI taking over should be aware of these issues.

As we have seen, those who believe in intuitionist mathematics think that the law of excluded middle presumes that mathematical objects are ostensibly real things existing independently from the human mind, which the intuitionist thinks is a fallacious claim. A lot of mathematicians sleep on the question of the ontology of mathematical objects, leaving it for the philosophers to discuss while they do not realise that the question impacts their work so much. If we were to move away from the 'accepted' idea that these objects exist and move towards mathematical intuitionism, then a number of our mathematical ideas change.

Appendix B will show off one of the key changes from Brouwer's intuitionism in the fact that every function is continuous and hence we cannot do the maths we

could before with discontinuous functions. Thus, these questions about whether mathematical objects are real or not are key due to their implications about how we should do maths. To summarise how intuitionists believe we gain mathematical knowledge, the key phrase to use is 'framework of perception'. Humans use mathematics in statistics, physics, economics and many other subjects that it is we use it to describe time, space and all the phenomena that occur within them both. As intuitionistic mathematics is just the use of mental constructs in order to show patterns in how we perceive things, it is one of the fundamental facets of the mind in percieving things scientifically. All of the maths used in sciences and social sciences is only us adapting these phenomena to the framework we know.

What is the alternative then? Mathematical Platonism seems to be the view that most mathematicians seem to sleepwalk into. This is the belief that mathematical objects exist independently of us. But is mathematical Platonism entailed in set theory? Not necessarily, for while Frege, whose work inspired multiple big names to research both set theory and type theory, believed that mathematical objects existed independently from human perception, Russell believed that the mathematical objects were nothing more than the logical constructions that we use to model them.

The ontology of mathematical objects is a discussion to be handled with care. We must give them the privilege of being non-physical due to the obvious differences between numbers and the objects in the physical world around us. Yet at the same time, we must make sure we do not come up with a philosophically untenable argument for justifying metaphysical objects that we don't necessarily perceive. There are many ways to perhaps justify that set theory does not entail mathematical Platonism, however intuitionists would still argue that using the law of excluded middle entails mathematical Platonism. Considering the fact that Diaconescu's theorem in ZFC states that the axiom of choice implies the law of excluded middle, intuitionists would not accept that you could have ZFC set theory without it entailing mathematical Platonism.

As stated in our discussion of the continuum hypothesis in Chapter 3, even if we let the law of the excluded middle runs its course, it still gives us a few discrepancies. Hence it may well be better to chuck it out. However, considering the sacrifice of discontinuous functions that we discussed in Chapter 4, we may wish to find a new way to model problems in probability and many other subjects which rely on the discontinuities of functions, as well as proofs in general which rely on the discontinuities of certain functions.

Hopefully these are all subjects which you, the reader, are now more interested in engaging with. Hopefully this encourages you to think more about the underlying machinery that is being used when we are reasoning through mathematical problems, whatever its basis and problems may be.

A Axiom of choice

The axiom of choice, when first posited, was met with a remarkable amount of controversy. As a matter of fact, it is still regarded with some controversy in some parts of the mathematical community. The Banach-Tarski Paradox is a showcase of the worst of the counter-intuitive nature of the axiom of choice, however there are ways in which it impacts measure theory as well which we will explore here.

The axiom of choice is equivalent to a number of important statements including the fact that every vector space has a basis, fundamental in algebra and Zorn's lemma, fundamental in analysis. For all the faults we are about to list, the axiom cannot be said to have no use.

As stated, when we first mentioned the axiom of choice, the key to its problems seem to be that unlike the rest of the axioms of ZFC, choice only posits that a set exists, but gives no indication as to what the set looks like (i.e. how the set is actually defined). It causes the following two counterintuitive results.

The Banach-Tarski paradox is a theorem which states that given a 3D ball of any volume, there is a way to cut up the ball into pieces and reassemble the pieces such that they can be assembled into two identical balls of the same volume. This is known as a paradox for the fact that it contradicts our basic geometric intuition. The axiom of choice is used in constructing sets before the reassembly of the sphere and while Banach-Tarski is provable in ZFC, it is not provable in just ZF (Zermelo-Frankel axioms without choice). Hence it seems that the axiom of choice is instrumental in the contradictory nature of this theorem.

Measure theory is the study of how we define the length of sets, areas under curves and volumes in mathematics. When initially deciding on some axioms with which we then proceed to uncover the rest of our theory with, specifically with regards to the length of sets in this case, three axioms that seem like they fit the notion of 'self-evident truths' may be the following:

(I) If we measure the length of an interval (one of $[a,b],(a,b),(a,b] or [a,b)$. This is a set which contains all the elements between a and b. If a or b is marked with a square bracket, it is a member of the set. Otherwise it is not a member of the set), then the length of these should be $b-a$.

(II) The length of the intervals $[a,b],(a,b),(a,b]$ or $[a,b)$ is equal to the length of the $[a+h,b+h],(a+h,b+h),(a+h,b+h]$ or $[a+h,b+h)$.

(III) If n sets are all pairwise disjoint (i.e. for two sets A and B, if $x \in A$ then $x \notin B$) then the lengths of the union is equal to the sum of the length of the individual sets.

The issue is, by constructing sets known as 'The Vitali sets' by using the axiom of choice, we can show that these axioms are inconsistent, as we get a situation where the length of $[0, 1]$, which by (I) should be 1, ends up being less than or equal to 0, an obvious contradiction which, yet again, happens due to the involvement of the axiom of choice.

The fact that the axiom of choice with function extensionality and propositional extensionality can be used to prove the law of excluded middle, which is controversial in its own way as we have documented time and again already in this essay.

B Additional remarks on intuitionistic mathematics

L.E.J. Brouwer managed to prove via the metatheory of his intuitionistic logic (i.e. using mathematical methods to analyse the syntax, semantics and proof theory of a formal language) that every function $f : \mathbb{R} \to \mathbb{R}$ is continuous, where $\mathbb{R}$ denotes the set of real numbers. To give the most simple (and, in many ways, the most effective definition of continuity), a function is continuous if you can draw all of it on an x-y plane without having to take your pencil off the paper (for those who want the formal definition of what it means to be a continuous mapping on the real numbers, given a function $f : \mathbb{R} \to \mathbb{R}$, it is continuous if $\forall\epsilon, \exists\delta$ such that whenever $x \in \mathbb{R}$, if $|x - c| < \delta$ then $|f(x) - f(c)| < \epsilon$, i.e. whenever x is close to c, then $f(x)$ is arbitrarily close to $f(c)$.) Brouwer's proof means that in his version of intuitionistic mathematics, functions which are not continuous (known as discontinuous functions) do not exist.

The idea that discontinuous functions do not exist is a foreign concept to most mathematicians and there are many valid arguments for stating that we should keep discontinuous functions as an entire concept. Of course, there are many fields of applied mathematics where discontinuous functions are applied such as probability and quantum mechanics. Even proofs in pure mathematics itself which rely on the discontinuity of certain functions would fail without the existence of discontinuous functions. While in mathematics, intuitionists could dismiss discontinuous functions easily by saying the concept of them entails the law of excluded middle, it is remarkably harder to justify having to rework the mathematical models that we have found so useful within the sciences.

There is a gaping rabbit hole here which I will have to direct us towards, however we will not be jumping down. The relationship between positions held in philosophy of mathematics and positions held in philosophy of science is something that is very much worth dissecting, to the point where teaching the two in conjunction would make for a fascinating undergraduate/postgraduate module. Hence I will quickly explain how an intuitionist may explain away the lack of existence of something integral to scientific fields now.

While one could hold that the scientific theories we currently have perfectly mirror the structure of the universe (the 'scientific realist' position), one could also hold that by looking at the history of science, we can see that mathematical models for predicting various phenomenon have changed massively over time, the unobservable scientific objects we use to describe the universe do not actually exist but we use the theories because currently they are the best explanation we have (the extreme 'scientific anti-realist' position known as instrumentalism).

A mathematical intuitionist could easily adopt the position of an instrumentalist and claim that we are only using discontinuous functions because in our

fundamentally incorrect frame of perception, they work in the scientific fields we then analyse with our frames of perception. Famously, Quine was very quick to suggest that mathematical rules should be subject to the same empirical scrutiny that the hypotheses of the physical sciences are. Using Quine's methodology, we could justify removing the law of excluded middle and just building a new mathematical model to fit in with our scientific theory. This could then go on to more accurately depict human perception of this scientific phenomenon.

There is an even simpler argument than this to make in favour of still keeping mathematical intuitionism, despite all we may lose for it. One could easily make the argument of 'so what?' in the face of people who think we should change our methods of doing mathematics for the benefit of science.

There are some key works in philosophy regarding arguments in analytic philosophy which seem to lie on the implicit argument that mathematics should work for the benefit of us then being able to apply it to our ability to observe the physical world. This idea that the purpose of studying mathematics is only so that it can then help us in studying science is a fallacious argument that I have never seen called out before, however I feel that this question *what is the purpose of doing mathematics?* is one which is not discussed enough, and so this argument is always seemingly allowed to pass with limited scrutiny. Hence an intuitionist can claim that mathematics is its own independent discipline which has no obligation to the physical sciences, so if discontinuous functions not existing is a problem for the sciences, then that's little more than tough luck!

C Glossary

When reading advanced texts while I was in sixth form, the one thing that annoyed me the most was the lack of a glossary in some introductory texts. These can be remarkably helpful for people trying to get into their academic discipline and should probably be normalised more, as it allows you to revise technical language without having to trudge through a lot of dense text.

Hence, I hope you enjoy this time-saver! If you're looking for a specific word and it is not here, then do not worry for it is likely that the word was not particularly important for your general understanding of the focus of this essay.

The terms are presented in alphabetical order:

Anti-realism – The philosophical position that certain objects do not exist independently from human perception. For example, a mathematical anti-realist believes that mathematical objects are just mental constructs.

Axiom – A self-evident truth used as the basis of a theory.

Axiom of Choice – An axiom from ZFC set theory (informally) stating that if we have any number of sets (including an infinite number!) then there exists a set which contains exactly one element of each set.

Axiom of Reducibility – The axiom in Russell's ramified type theory stating that for every which states that for each type of a class C, there is an impredicative class I with the same members as C.

Cardinality – The amount of elements in a set.

Constructivism – A position in philosophy of mathematics which states that a mathematical statement is true if we can construct a proof of it.

Continuous Function – A function which you can draw in an x-y plane without taking your pen off of the paper (more formally, $f : \mathbb{R} \to \mathbb{R}$, it is continuous if $\forall\epsilon, \exists\delta$ such that whenever $x \in \mathbb{R}$, if $|x - c| < \delta$ then $|f(x) - f(c)| < \epsilon$).

Continuum Hypothesis – The hypothesis which states that there are no sets whose cardinality's value is greater than the cardinality of the natural numbers and less than the cardinality of the real numbers.

Curry-Howard Correspondence – The one-to-one correspondence between programs in the lambda calculus (a model of computation) and proofs in natural deduction systems in intuitionistic logic. Also known as the Curry-Howard isomorphism.

Dependent Type Theory – A variation of intuitionistic type theory which introduces dependent types such as Π-types and Σ-types.

Diaconescu's theorem – The axiom of choice, propositional extensionality and function extensionality imply the law of excluded middle.

Element – An object that belongs to a set (the object itself could be a set).

Epistemology – The philosophical study of knowledge and how we gain knowledge.

First-order logic – A formal language with the same structure as propositional logic, however with addition rules to accommodate for additional variables and quantifiers.

Formalising – Expressing a sentence from natural language/ normal mathematics in terms of a formal language (e.g. formalising mathematics in Lean consists of us writing mathematical statements in the terms of dependent type theory).

Formal language – A language which consists of sentences formed by using letters to denote atomic sentences as well as the rules from the syntax of the language (e.g. connectives in propositional logic) to create a 'well-formed formula' from the language.

Functional Programming – Programs are constructed by applying functions and composing functions together (i.e. applying the result of one function to another function).

Function – A mapping of elements from one set (known as the domain) to elements in the same/another set (known as the codomain).

Function Type – A mapping of individuals from one type (known as the domain) to individuals in the same/another type (known as the codomain).

Gödel's Incompleteness theorems – A theorem which states that for certain theories, there are statements which are consistent with the theories but not provable from the axioms of the theory.

Homotopy Type Theory – A recent new version of intuitionistic type theory which draws similarities from homotopy theory in algebraic topology and shows their similarities with intuitionistic type theory.

Impredicative – An object defined from a collection of objects e.g. "the grumpiest maths students" is an object defined from comparing every maths student.

Individual – A mathematical object that is a member of a type (it could be a type itself)

Intuitionism – The position in philosophy of mathematics that states we should reject the law of excluded middle as it entails mathematical realism.

Lambda Calculus – A language for computing functions which uses lambda abstraction and lambda application to express terms.

Law of Excluded Middle – The sentence in classical logic that states that either every sentence or its negation is true (written formally as $P \vee \neg P$

Logic – The study of human reasoning by creating formal languages to study the syntax (the rules of writing out the terms of the language), semantics (how we can tell a sentence in a language is true or false) and proof theory (how we formally prove a conclusion from a number of hypotheses of a language) of language/reasoning.

Logicism – The position in philosophy of mathematics which states that mathematics is just an application of logic and every mathematical statement can be rewritten as a logical statement.

Naïve set theory – Set theory being used without any axioms.

Natural number – 1,2, ... , i.e. all of the whole numbers above 0 (in some literature, mathematicians will define 0 to be a member of the natural numbers).

Ontology – The philosophical study of 'being' and the what it means for certain objects 'to be' and exist.

Platonism – The belief that all objects have an ideal form which exists on a greater plane of being than our reality.

Predicative – An object that can be defined without having to refer to a group of objects.

Program – A set of instructions given to a computer for the computer to then run and give a result when given an input.

Proof – Rigorous and true steps taken from a hypothesis to a conclusion in order to show that the conclusion is true when the hypotheses are also true.

Proof by contradiction – A proof that starts by assuming the negation of what you are trying to prove and deriving a contradiction, effectively showing that it is impossible for the initial statement to not be true.

Propositional logic – A formal language using the 5 main logical connectives $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$, parentheses, and letters to denote sentences.

Ramified Type Theory – Russell's type theory, which included a hierarchy of types to denote whether a type was a collection of individuals, types, types of types, etc.

Real numbers – Numbers that can be written as ratios (e.g. 2, 5/2, 198/237) and numbers that cannot be written as ratios (π, e).

Realism – The belief that objects around us exist independently from human perception.

Set – A collection of objects.

Set theory – The study of sets.

Type – A collection of objects ascribed a type, effectively a set but more specific about the contents of the set.

Type theory – The study of types.

ZFC set theory – Zermelo-Fraenkel-Choice set theory. Uses each of the ZF axioms and includes choice.

ZF set theory – Zermelo-Fraenkel set theory, using every axiom used in ZFC except without the axiom of choice.

D Further Reading

For those wanting to explore first-order logic and predicate logic, then I would recommend both Logic and Structure, 5th edition by Dirk van Dalen, Springer Verlag, 2013 as well as Language, Proof and Logic, Jon Barwise and John Etchemendy, CSLI Publications, 2002. The latter is much better as an introductory text so I would recommend starting with that first.

For more set theory, I would recommend checking Set Theory: The Third Millennium Edition, revised and expanded, 3rd edition by Thomas Jech, Springer Verlag, 2006. It starts with the ten axioms stated here and expands on them further.

For more on the lambda calculus and combintory logic, I would recommend checking out Lambda-Calculus and Combinators, an Introduction by Hindley and Seldin, Cambridge University Press, 2008. Before diving into it, I would recommend knowing a bit about computability theory and metalogic in general (the latter can be found in Logic and Structure, mentioned above.)

For philosophy of mathematics, Thinking about mathematics by Stewart Shapiro, Oxford University Press, 2000 is a great place to start, with great further reading suggestions in there as well for anyone wishing to interrogate specific issues deeper.

For those wanting to know more about Intuitionistic Type Theory, then Per Martin-Löf's 1980 essay of the same name is a great starting point. Again, I would recommend having a background knowledge of logic and metatheory before diving in.

For learning Lean itself, if you are a forward-thinking mathematician of the future, there are a number of places you can go. Theorem proving in Lean is where I have learned how to use it, it will take you a while to get through and having a background in the lambda calculus/ logic is what helped me crack it (mainly in terms of understanding other people's code.) A link to that can be found here: https://leanprover.github.io/theorem_proving_in_lean/.

There are, of course, more hands-on approaches for those who like to dive right in. http://wwwf.imperial.ac.uk/~buzzard/xena/ is the main page for all things Xena project. Located here are a number of games and tutorials which will teach you how to program in Lean, as well as download links for Lean and a useful document by Mario Carniero which explains even more of the technical nature of the type theory used in Lean.

Some sixth-formers and undergrads read one introductory document (like this!) and sometimes get it in their head that they are suddenly an expert on the subject. This is very much not how it works so I would recommend exploring

the above! Even with all the above under your belt, you still won't be an expert. You'll be slightly more specialised and able to blag that you're an expert to your friends though (I refer to this as the Seemungalian approach).

E References

Avigard, De Moura, Kong, Theorem Proving in Lean, Release 3.18.4, 2020, chapters 1-4, 7, 11

Jesse Michael Han and Floris van Doorn. 2020. A Formal Proof of the Independence of the Continuum Hypothesis. In Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '20), January 20–21, 2020, New Orleans, LA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372885. 3373826

Hindley and Seldin, Lambda-Calculus and Combinators, an Introduction, Cambridge University Press, 2008, chapters 1, 10

Jech, Set Theory: The Third Millennium Edition, revised and expanded, 3rd edition, Springer Verlag, 2006, chapter 1

Martin-Löf, Intuitionistic Type Theory, Bibliopolis, edizioni di filosofia e scienze, 1980

Rorvig, Number Theorist Fears All Published Maths Is Wrong, https://www.vice.com/en_uk/article/8xwm54/number-theorist-fears-all-published-maths-is-wrong

Shapiro, Thinking about mathematics, Oxford University Press, 2000, chapters 5,7