BULLETIN (New Series) OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 61, Number 2, April 2024, Pages 257–270
https://doi.org/10.1090/bull/1830
Article electronically published on February 15, 2024

STRANGE NEW UNIVERSES: PROOF ASSISTANTS AND SYNTHETIC FOUNDATIONS

MICHAEL SHULMAN

Abstract. Existing computer programs called proof assistants can verify the correctness of mathematical proofs but their specialized proof languages present a barrier to entry for many mathematicians. Large language models have the potential to lower this barrier, enabling mathematicians to interact with proof assistants in a more familiar vernacular. Among other advantages, this may allow mathematicians to explore radically new kinds of mathematics using an LLM-powered proof assistant to train their intuitions as well as ensure their arguments are correct. Existing proof assistants have already played this role for fields such as homotopy type theory.

"Out of nothing I have created a strange new universe."
– János Bolyai, one of the inventors of non-Euclidean geometry

1. Introduction

My most upvoted post on the question-and-answer website MathOverflow is an answer to the question "Why doesn't mathematics collapse even though humans quite often make mistakes in their proofs?". In my answer [Shu20], I wrote:

> [The] fundamental content of mathematics is *ideas* and *understanding*, not only proofs. If mathematics were done by computers that mindlessly searched for theorems and proof but sometimes made mistakes in their proofs, then I expect that it would collapse. But usually when a human mathematician proves a theorem, they do it by achieving some new understanding or idea, and usually that idea is "correct" even if the first proof given involving it is not.

At the time in 2020, the idea of a computer "mindlessly searching for theorems and proof, but sometimes making mistakes" seemed fanciful. But in 2023, Large Language Models (LLMs) such as ChatGPT can do exactly that. Currently, in fact, their mistakes are quite frequent. These systems will continue to improve, but at root they are just word-prediction algorithms, which cannot understand ideas or check a proof step-by-step like a human. (This is what I meant by "mindlessly".)

Of course, human mathematicians also make mistakes. I don't know whether an LLM will one day "check" a proof as reliably as a human. But I also don't think this is a very relevant question, because computer programs called *proof assistants* can already verify proofs more reliably than either humans or LLMs. Proof assistants are not yet used widely, mainly because they require a specialized "coding" of proofs

Received by the editors August 12, 2023.
2020 *Mathematics Subject Classification.* Primary 68V20.
This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0009.

©2024 by Michael Shulman

(see Section 2). But this is likely not a permanent impediment, because LLMs excel at mediating such human-computer interaction.

Thus, I believe that future mathematics software will combine the ease of interaction of an LLM with the near-absolute trustworthiness of a proof assistant. Development of such *autoformalizers* is already under way (see Section 3). This would answer the original question reassuringly: if formalized proof were as easy as $\LaTeX$, mistakes could be vanishingly rare. But it could also make mathematics done by a computer just as trustworthy as mathematics done by a human.

So will human mathematicians become unnecessary? Considering the effects of other technological advances throughout history, I doubt it. Furthermore, as noted above, mathematics is fundamentally about ideas and understanding—*human* understanding, not whatever sort of "understanding" an LLM might have.

However, I do believe LLMs may *change* what human mathematicians spend our time doing, perhaps drastically. A present-day mathematician knows the fundamentals of arithmetic, but no longer needs to use tables of logarithms. Similarly, we can imagine that future mathematicians might still know the fundamentals of logic, but no longer need the same technical facility of proof construction that we rely on today. Our conceptual understanding need not be impacted; instead the technology can free us to put *more* time and effort into understanding. In particular, just as a present-day mathematician can exploit a computer's *calculational* ability to explore *numerical* realms undreamed-of decades ago, these imagined future mathematicians may exploit a computer's *logical* ability to explore *conceptual* realms undreamed-of today.

I find this last possibility the most exciting. As evidence that such realms exist, I offer *homotopy type theory and univalent foundations* (Section 4). Barely a decade old, this field proposes a dramatic change to the set-theoretic perspective that has reigned supreme in mathematics for close to a century. Rather than sets, in homotopy type theory we build mathematical structures out of a primitive notion of *space*, a.k.a. *homotopy type*, a.k.a. *∞-groupoid*. This has some advantages for existing mathematics, such as automatic isomorphism-invariance, but it also opens up entirely *new kinds* of mathematics, such as *synthetic homotopy theory* (Section 5).

The rules of logic in homotopy type theory are similar to, but subtly different than, those of traditional mathematics. Thus, it is easy to make mistakes by forgetting these differences. For this reason (among others), today's practitioners of homotopy type theory have relied heavily on current proof assistants, both to verify proofs and to train intuitions. This is a marked departure from the way proof assistants have so far been used in most other areas of mathematics, where theorems are generally proven by hand first and only later formalized in a computer.[1]

I believe that homotopy type theory could not have enjoyed its current level of success if proof assistants were less capable than they are today. What new kinds of mathematics, then, may be waiting for us to explore with the proof assistants of tomorrow? In Section 6, I will discuss one possibility suggested by the current frontiers of homotopy type theory, namely *modal* type theories; but I expect this

[1]Similar uses of proof assistants are also starting to appear in other fields, however, such as the Liquid Tensor Experiment [Sch21], about which Scholze wrote, "here we have witnessed an experiment where the proof assistant has actually assisted in understanding the proof."

barely scratches the surface.[2] As mathematicians are liberated from calculation and tedium to spend more energy on ideas and understanding, who knows what strange new universes we will create?

2. The structure of a proof assistant

At the core of a proof assistant is a program called the *kernel* that checks whether a proof is correct. The meaning of "correct" is determined by a so-called *foundational theory*, which here simply means a formal system that is expressive enough that "all of mathematics" can be encoded into it.[3]

The best known foundational theory is ZFC/FOL: Zermelo–Fraenkel set theory (with the axiom of Choice) expressed in first-order logic. Here FOL supplies rules for constructing and proving things called "propositions" (statements), while ZFC supplies axioms about things called "sets" that are used in such proofs. The proof assistant Mizar[4] has a kernel using a theory like ZFC/FOL.

Another class of foundational theories is the *dependent type theories* in the tradition of Martin-Löf [ML75, ML84]. Rather than separate layers of propositions and sets, these theories have one layer with rules for constructing and inhabiting things called *types*. Some types are treated like sets and used to build mathematical structures, while other types are treated like propositions and used to prove things. Remarkably, the same rules for types specialize to the basic operations of both sets and propositions; this is called *propositions-as-types* or the *Curry–Howard correspondence*. For instance, the cartesian product set $A \times B$ and the conjoined proposition "P and Q" are unified by the notion of *product type*. Many proof assistants like Coq,[5] Agda,[6] and Lean[7] have kernels built on dependent type theory.[8]

In general, the kernel of a proof assistant takes as input a proof written in the language of the foundational theory, and checks whether it correctly follows the rules. Such a formalized proof closely resembles a computer program, and a kernel is analogous to a compiler. However, because the kernel must be trusted implicitly, the foundational theory is generally as simple as possible, with hardly any bells and whistles for convenience; thus it is most like a *machine code* for mathematics. For example, the representation of "$1 + 1 = 2$" in foundational dependent type theory may look like $\mathsf{Id}\,\mathbb{N}\,((\lambda x.\lambda y.\mathsf{rec}\,\mathbb{N}\,x\,(u.v.\mathsf{S}\,v)\,y)\,(\mathsf{S}\,0)\,(\mathsf{S}\,0))\,(\mathsf{S}\,\mathsf{S}\,0)$.

The spartan nature of the foundational theory makes it tedious for a human to use directly, so proof assistants also incorporate an *elaborator*. This is like a compiler that translates "programs" (here, constructions and proofs) from a "high-level language" into "machine code" (the foundational theory) that can be "executed"

[2]I have used several examples from my own work in this paper, primarily because in such cases I can speak authoritatively not only about the results, but about the processes that led to them and the experiences of proving them. I do not intend to downplay or minimize the contributions of others; think of it as a personal story rather than a comprehensive survey.

[3]This property is formally similar to those of Turing-complete programming languages and NP-complete problems, so a foundational theory might also be called "mathematics-complete".

[4]`http://mizar.org/`

[5]`http://coq.inria.fr/`

[6]`https://wiki.portal.chalmers.se/agda/pmwiki.php`

[7]`https://leanprover.github.io/`

[8]Other foundational theories include Lawvere's ETCS/FOL [Law05] (not used by any proof assistant that I know of) and the *simple type theories* (higher-order logics) used by proof assistants such as HOL (`https://hol-theorem-prover.org/`) and Isabelle (`https://isabelle.in.tum.de/`).

(verified by the kernel). The high-level language of a proof assistant is called its *vernacular*; it implements features with names like "implicit coercions", "higher-order unification", "namespaces", "mixfix notations", and so on, which allow vernacular proofs to look more like pencil-and-paper mathematics. For instance, in a vernacular we can write the above expression as `1 + 1 = 2`.

In principle, the vernacular could be any sufficiently expressive formal system. However, decades of experience have led most designers of proof assistants to conclude that type theories make better vernaculars than set theories do; a nice explanation was given by Bauer [Bau20]. (This is one reason that most newer proof assistants take the foundational theory to be a type theory as well.)

The vernacular improves on the foundational theory, but today's vernaculars still require more details than a mathematician would ordinarily write on paper. Thus, formalizing mathematics in a proof assistant is still a more substantial undertaking than writing it for human readers. However, in certain restricted domains, modern vernaculars incorporate *automation*,[9] which can generate a proof by applying decision algorithms or searching a library for applicable lemmas. For example, many proof assistants can automatically prove identities that hold in any commutative ring, such as $(3x^2 - 2yz)(xy + 4z) = y(3x^3 - 8z^2) - 2xz(y^2 - 6x)$. Formal proofs using automation can be even *more* concise than those written on paper.

The flip side is that such "proofs by automation" often convey little *understanding* of why the statement is true. Sometimes this is no loss: e.g., most mathematicians could prove a commutative ring identity themselves, so automating it just saves effort. But other times an automated proof can feel like "magic" and leave a human reader feeling unsatisfied, and this is likely to become more common as automation becomes more powerful. And, truth be told, even less-automated vernacular proofs are more difficult for a human to understand than traditional proofs in a human language. Thus, published mathematics using a proof assistant often has a dual existence: a traditional paper for humans to read and a parallel computer-verified formalization.

Reading such mathematics is a slightly different experience from reading unformalized mathematics. On one hand, the "human-readable" proofs still often contain fewer details than otherwise, on the theory that these were mainly for convincing the reader of the *correctness* of the proof, which is now unnecessary. The authors thus tend to concentrate on those parts of the proofs that they view as central to *understanding*, expecting the reader to trust the proof assistant regarding the more tedious or calculational parts. On the other hand, while a human reader no longer needs to check every proof, there is now a different obligation: someone must check that the formalization actually formalizes what the author claims. In addition to executing the proof assistant on the author's code, this requires checking that the *statements* being proved in the vernacular match those written in the paper, and that the formalized proofs do not use any hidden assumptions or axioms. (As a referee of formalized papers, I have encountered errors of both sorts.)

[9]Sometimes automation is considered instead part of the *meta-language* or *tactic language*, a third layer distinguished from the vernacular.

3. Autoformalization and auto-informalization

Using the terminology just introduced, the suggested integration of LLMs with proof assistants would bring the "vernacular" closer to ordinary mathematical writing, perhaps eventually eliminating the parallel existence of paper and formalized mathematics. Such a vernacular would be less precisely specified than those of today; in a sense, it would consist entirely of "automation", with the LLM translating a human-readable proof directly into the foundational theory. Put differently, the LLM would function as a more powerful and flexible elaborator, which does not require its input to be so rigidly specified. This is known as *autoformalization*, and is already being developed [JSS+22, AGG+22, JWZ+23].

In an integrated autoformalization system, the unaltered trustworthiness of the kernel would mean that any "hallucinations" of the LLM would not affect the correctness of the proof. One would still need to ensure that the formalized "theorem" matches the claim, which might still require a more precisely specified vernacular alongside the LLM. But this small amount of extra work could easily be offset by the benefits of autoformalization, along with other improved kinds of automation.

So much for correctness; what about the more "fundamental content" of ideas and understanding? An autoformalizing proof assistant would reduce one way in which formalized mathematics challenges understanding: the vernacular "code" could be directly understood by a human mathematician. But we can also expect advances in automated proof *generation*, which also challenges understanding: how can a human read and understand a proof that was generated by a computer directly in the foundational theory without any vernacular version at all? For this reason I believe we will also see a rise of *auto-informalization*: LLM-powered software for translating *back* from proofs in the foundational theory to human-readable ones. The degree of detail could even be customized to the reader: students may require all details spelled out, while experts may prefer a concise summary.

In summary, I believe humans will always remain in charge of mathematics, with a focus on ideas and understanding, not just proofs and correctness. But in addition to making today's mathematics easier, faster, and more reliable, I believe the enhanced proof assistants of the future may enable us to achieve new results and understandings that would previously have been prohibitively difficult.

One possibility in this regard is *quantitatively* different mathematics. Some proofs are just too long, with too many cases or ideas, for any human mathematician to completely understand, but a proof assistant can still guarantee their correctness. For instance, the Appel–Haken proof of the four color theorem, which involves hundreds of cases generated and checked by a computer, has now been formalized in a proof assistant [Gon08], ensuring that the case-checking software has no bugs that imperil the proof. Likewise, Hales's proof of the Kepler conjecture was so long and complicated that the human referees assigned to it gave up after several years of work, but it has now been verified in a proof assistant [HAB+17]. However, the limitations of today's proof assistants mean that so far, all such projects have required many years of work by large teams. I believe that with the proof assistants of the future, we can expect such proofs to become more commonplace, and within the reach of solitary mathematicians.

A different possibility, which I will spend the rest of the paper discussing, is the accessibility of *qualitatively* different mathematics.

4. Homotopy type theory and univalent foundations

I said previously that a foundational theory must be sufficiently expressive that "all of mathematics" can be encoded into it. The quotation marks are necessary, however, because mathematics is continually growing, and its interaction with foundations is a two-way street. A foundational theory must, by definition, be able to encode the mathematics extant when it is formulated; but it may then suggest new principles and techniques, changing the face of future mathematics. This was true historically for ZFC/FOL, and may likewise be the case for more recent theories.

Of these, a particularly interesting one is *homotopy type theory and univalent foundations* (HoTT/UF).[10] This refers to a new class of dependent type theories, based on insights of Awodey and Warren, Voevodsky, and others in the late 2000s. I mentioned in Section 2 that dependent type theories give rules for constructing and inhabiting *types*, and that traditionally some types are treated like propositions while others are treated like sets. In HoTT/UF, these two classes of types are the bottom two rungs on an infinite ladder, called respectively (-1)-types and 0-types.[11] There are then also 1-types, 2-types, and so on, while arbitrary types may not have any finite dimension.

This stratification arises from the behavior of equality. In ZFC/FOL, if x and y are sets, then $x = y$ is a proposition. In dependent type theory, since both propositions and sets are types, we say more generally that if x and y are elements of any type A, then $x = y$ is also a *type*. The rules governing this type incarnate the Leibnizian principle of "indiscernibility of identicals". If A is a 0-type (set-like), we expect $x = y$ to be a (-1)-type (proposition-like); formally, this means it has at most one element. More generally, we define A to be an $(n+1)$-type if for any x and y in A, the type $x = y$ is an n-type.

An example of a 1-type in HoTT/UF is the type of groups.[12] If G and G' are two elements of this type, i.e., two groups, then $G = G'$ is a 0-type (set-like), whose elements turn out to be the *group isomorphisms* from G to G'. More precisely, a group is a tuple $(G_0, m, e, i, \alpha, \lambda, \rho, \iota)$ where G_0 is a 0-type, $m : G_0 \times G_0 \to G_0$ is a multiplication, e is an identity, $i : G_0 \to G_0$ is the inversion, and $\alpha, \lambda, \rho, \iota$ are witnesses of the group axioms (i.e., elements of the corresponding proposition-like types). The compositional nature of equality means that an element of the equality type $(G_0, m, e, i, \alpha, \lambda, \rho, \iota) = (G_0', m', e', i', \alpha', \lambda', \rho', \iota')$ consists of componentwise equalities $G_0 = G_0'$, $m = m'$, $e = e'$, and so on.

Now, a central principle of HoTT/UF, known as Voevodsky's *univalence axiom*, says that the elements of $G_0 = G_0'$ are *bijections* between the 0-types G_0 and G_0'. Then since m and m' don't belong to the same type ($G_0 \times G_0 \to G_0$ versus $G_0' \times G_0' \to G_0'$), an equality $m = m'$ must actually be a "dependent equality" parametrized by the *specified* bijection between G_0 and G_0', and this turns out to say precisely that this bijection preserves the group multiplication. Likewise, the other equalities say that the bijection preserves identity elements and inverses (automatic

[10] While the slash in "ZFC/FOL" connects a theory to its logical substrate, the slash in "HoTT/UF" connects two phrases with closely related but nonidentical meanings. But the difference between HoTT and UF is not uniformly agreed-upon, and irrelevant for the present.

[11] Technically, there is an even lower rung at -2, but it contains only the one-point type.

[12] More precisely, the type of groups with cardinality smaller than some fixed bound. Due to Russellian paradoxes there is no type of literally *all* groups.

in the case of groups) as well as the axioms (always automatic, because they are elements of (-1)-types). Thus, equalities of groups are group isomorphisms.

In ZFC/FOL, groups and group isomorphisms form a *groupoid*, meaning that isomorphisms can be composed and inverted. Equality types automatically have analogous structure, so 1-types are a different way to represent groupoids. More generally, n-types behave like "n-groupoids", and arbitrary types behave like "∞-groupoids". For example, the type of categories is a 2-type. If $\mathcal{C}$ and $\mathcal{D}$ are categories, then $\mathcal{C} = \mathcal{D}$ is the type of *equivalences of categories* from $\mathcal{C}$ to $\mathcal{D}$, which is a 1-type: its equality types consist of *natural isomorphisms*.

Prior to HoTT/UF, type theorists tended to assume that all types are set-like. But this is not provable from the rules of equality in type theory, so it was taken as an extra axiom. The original insight of HoTT/UF, therefore, is that the philosophically founded Leibnizian concept of equality *naturally* leads to higher types, if we simply refrain from closing off that possibility by fiat.

Since most of present-day mathematics deals with structures built from sets, its formalization in HoTT/UF primarily uses 0-types and (-1)-types. But higher types do sometimes appear, particularly when notions from category theory are present implicitly or explicitly. When this occurs, its practical upshot is to ensure *isomorphism-invariance*. For instance, because $G = G'$ is the 0-type of group isomorphisms from G to G', the principle of "substitution of equals for equals" implies that if G and G' are isomorphic, any true statement about G is also true about G'. In this way, HoTT/UF directly enforces isomorphism-invariance, and also equivalence-invariance for categories and higher structures.

Thus, although all of present-day mathematics can be coded into both ZFC/FOL and HoTT/UF, the two encodings are rather different. For instance, in ZFC/FOL we define a groupoid as a collection of sets equipped with algebraic structure, whereas in HoTT/UF we represent these same data by a single primitive object (a 1-type). Likewise, some statements that require tedious proofs when coded in ZFC/FOL, such as transport or invariance along isomorphisms, are automatic in HoTT/UF. In this way, HoTT/UF is closer to informal mathematics than ZFC/FOL is, since isomorphism-invariance is commonly used implicitly without proof.

On the other hand, due to the influence of ZFC/FOL, present-day informal mathematics calls all collections "sets", and treats all equalities as propositions. Thus, when encoding it into HoTT/UF, we have to decide whether each use of "set" should be "0-type" or "n-type" for some $n > 0$ (including ∞); and in the latter case the translation can be highly nontrivial.

Of course, just because most mathematicians today assume that all collections are sets and all equalities are propositions doesn't mean that that need always be the case. Indeed, the book *Homotopy Type Theory—Univalent Foundations of Mathematics* [Uni13] proposed an informal style of mathematics closer to HoTT/UF, which is now common in the HoTT/UF community. We might call this *univalent mathematics*, in contrast to traditional *set-based mathematics*. Thus, HoTT/UF could serve as a foundation for mathematics in exactly the same way as ZFC/FOL, with an informal mathematical vernacular that could be—but may or may not ever actually be—completely translated into the formal theory.

However, at present a much larger proportion of univalent mathematics *is* formalized in a proof assistant than is usual in set-based mathematics. One reason is that the HoTT/UF community has a large overlap with computer science and proof formalization.[13] But in addition, mathematics involving n-types for $n > 0$ is *harder to get right* than set-level mathematics (regardless of whether the latter is done in HoTT/UF or in ZFC/FOL), so there is more value in computer formalizations.

Why is this? One reason is that most modern mathematicians were originally trained in set-based mathematics. But it must also be admitted that the behavior of n-types for $n > 0$ is intrinsically further from our everyday experience: when speaking of trees, horses, or taxicabs, equality really is just a proposition. Accordingly, it is easy to make mistakes by assuming, wrongly, that univalent equality behaves like set-theoretic equality, without even noticing the assumption.

As an example, suppose $f : X \to X$ is a function that is *idempotent*, i.e., $f(f(x)) = f(x)$ for all $x \in X$. In set-based mathematics, we can *split* f, meaning to write $f = s \circ r$ for two functions $r : X \to Y$ and $s : Y \to X$ such that $r(s(y)) = y$ for all $y \in Y$. Namely, we define $Y = \{ x \in X \mid f(x) = x \}$, with $s(y) = y$ and $r(x) = f(x)$. But in univalent mathematics, $f(x) = x$ can contain data that is forgotten by this s, so that $r(s(y)) = y$ no longer holds. And indeed, not every idempotent function in univalent mathematics *can* be split; we require an extra "coherence" condition on the equality datum $f(f(x)) = f(x)$.

One way to develop intuition for HoTT/UF is by translating its concepts into those of ZFC/FOL. Indeed, Voevodsky [KL21] constructed a *model* of HoTT/UF in ZFC/FOL, in which the types of the former are interpreted by the ∞-groupoids of the latter, and in [Shu19] I generalized this to all ∞-toposes in the sense of [Lur09].[14] Thus ∞-toposes in ZFC/FOL can teach us something of how the types in HoTT/UF behave, and sometimes even yield directly translatable proofs. For example, the (possibly surprising) fact that *one* extra coherence condition is necessary and sufficient to split an idempotent was first shown for ∞-toposes by Lurie [Lur14]. When I was later asked the analogous question in HoTT/UF, I went immediately to Lurie's work and came away not only with the theorem that I should expect to hold, but a proof idea that I was able to use with only small changes [Shu16].

However, this approach has limitations. One is that these models use fairly intricate homotopy-theoretic machinery, requiring significantly more background to understand. Another is that they are not "complete": there are statements true in every ∞-topos but not provable in HoTT/UF.[15] More importantly, even if such a statement is provable in HoTT/UF, as often as not the ∞-topos proof does not translate and a new proof must be found. For instance, the Blakers–Massey theorem in HoTT/UF [FFLL16] was unable to follow any classical proof for ∞-toposes.

Thus, a more accessible and effective way to construct trustworthy proofs in HoTT/UF is to use a proof assistant. Indeed, in the HoTT/UF community it is

[13]There are many possible reasons for this. For instance, since HoTT/UF uses dependent type theory, it is in some ways easier to learn for computer scientists, who tend to already be familiar with other type theories, than for mathematicians, who tend to be more set-theoretically trained. Computer scientists are also more open to the invention of new programming languages, whereas mathematicians tend to think of the "foundations" of mathematics as fixed and unchanging. And, of course, there is feedback from the other reason mentioned in the text: the additional benefits of computer formalization for HoTT/UF mean that it tends to attract mathematicians with a prior interest in formalization, and also that homotopy type theorists tend to *learn* some computer formalization even if they had no prior experience with it.

[14]Conversely, [Uni13, Chapter 10] constructs a model of ZFC/FOL inside HoTT/UF. Thus the two theories are, at least formally, equally adequate as foundations for mathematics.

[15]One can circumvent this by generalizing from Grothendieck and Lurie ∞-toposes to a kind of "elementary" ones, but then the statement becomes so tautological as to be unhelpful.

not uncommon to *develop* new mathematics with a proof assistant, rather than merely formalizing a proof already written on paper. The proof assistant provides immediate feedback about the correctness of each step and the current status of the proof, validating or rejecting the mathematician's understanding, helping to develop his or her intuition, and ensuring correctness dynamically. Afterwards, the formal proof can potentially be "informalized" into a more human-readable one; much of the informal mathematics in [Uni13] already arose from this process.

With current proof assistants, such an approach is currently unpalatable in most fields of mathematics. But the tradeoffs are different when working with higher types in HoTT/UF: partly because the benefits are greater, as described above; and partly because this sort of mathematics (which I will discuss further in Section 5) is currently still fairly close to the foundations, making the additional burden of formalization relatively small. As the technology of proof assistants becomes more accessible, I believe this way of doing mathematics will likewise become more common. This, in turn, may significantly lower the barrier to entry for mathematicians to work in novel foundational theories such as HoTT/UF.

5. Synthetic homotopy theory

Given the additional difficulty of doing mathematics in HoTT/UF, it is natural to wonder why we would bother at all. One reason, mentioned previously, is automatic isomorphism-invariance. But a more interesting reason is that there is *new* mathematics we can do in HoTT/UF that is essentially impossible in ZFC/FOL.

Recall that the types of HoTT/UF act like ∞-groupoids in ZFC/FOL. Classically, ∞-groupoids are often the "homotopy types" of topological spaces, remembering the points, paths, deformations, and higher deformations, but not the strict topology. The coincidence of the word "type" is fortuitous, since many such homotopy types in ZFC/FOL have corresponding types in HoTT/UF.

For instance, HoTT/UF has a type called S^1 that behaves like the homotopy type of a circle: it has a basepoint b and a nontrivial "loop" l in the type $b = b$, representing the "path" or "isomorphism" that goes around the circle once. There is also a univalent "fundamental group" π_1, which measures the number of ways we can "wind around" the equalities in a type. And we can prove, as in classical homotopy theory, that $\pi_1(S^1) \cong \mathbb{Z}$: a path drawn on a circle is characterized, up to deformation, by an integral winding number.

However, the proof of this fact in HoTT/UF is different than the classical one, though related: both use a *universal cover* of S^1, but the univalent proof defines this cover using the "induction principle" of S^1 as a "higher inductive type", plus the univalence axiom. These words are explained in [Uni13] and elsewhere; here I only want to note that while the statement is similar to one from set-based mathematics, it is in fact a different statement about a different object and has a different proof. My first proof of the HoTT/UF result [Shu11] (which was developed using a proof assistant and only afterwards "informalized") emphasized its similarities to the classical one. But a deeper analysis by Licata [LS13] uncovered an underlying general method, now called "encode-decode", that has no real classical counterpart. In particular, when this proof is interpreted in Voevodsky's model, it yields a proof about ∞-groupoids in ZFC/FOL; but it is a new proof, albeit of an old theorem.

Since the analogous set-based result belongs to homotopy theory, this sort of proof in HoTT/UF is known as *synthetic homotopy theory*, in contrast to classical *analytic* homotopy theory in the same way that Euclid's *synthetic geometry* contrasts with the analytic notion of coordinatized geometry. Despite its similarities and debt to classical homotopy theory, synthetic homotopy theory is a new branch of mathematics, which is essentially invisible to ZFC/FOL: the statement that "the fundamental group of the higher inductive type S^1 is $\mathbb{Z}$" doesn't even mean anything in ZFC/FOL, since it has no higher types.[16]

I find this idea quite exciting for a number of reasons. One is that, just as Euclid's geometry admits "nonstandard" models, a proof in synthetic homotopy theory can automatically be translated, not only to a proof about ∞-groupoids, but to a proof about objects of any ∞-topos. These models may seem bizarre at first; but, like the "strange new universes" of non-Euclidean geometry, we soon discover that they have their own beauty and a wide range of applications. Concepts from higher category theory and homotopy theory are increasingly appearing in many areas of mathematics, and often this can be described as working inside a particular ∞-topos specific to that area, such as sheaves on a space or a site. Synthetic homotopy theory provides a way to prove theorems in such ∞-toposes using language that is no more complicated than that of spaces in ordinary homotopy theory. From this perspective, HoTT/UF can be thought of as a "domain-specific language" that we can use for reasoning about ∞-toposes even if we choose ZFC/FOL as "the" foundation of mathematics.[17] I believe that one day, this sort of foundational transfer will also be incorporated into proof assistants.

Another reason I find synthetic homotopy theory exciting is that, compared with classical homotopy theory, it is very close to the foundations of mathematics, and yet it includes the deepest problems of homotopy theory. By the former I mean that after setting up the foundational theory of HoTT/UF, it only takes a few lines to define spheres S^n and homotopy groups π_k. For instance, the circle S^1 is simply defined as the type *freely generated* by the basepoint b and the loop l. By contrast, to define the analogous objects in ZFC/FOL requires first defining real numbers and proving many of their properties in order to construct spheres and homotopies. (Of course, ZFC/FOL has different objects that are close to *its* foundations, such as the von Neumann ordinals $\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \dots$.)

However, despite the simplicity of spheres and homotopy groups in HoTT/UF, they immediately give rise to a fascinating sequence of invariants: the groups $\pi_k(S^n)$, which are called the *higher homotopy groups of spheres*. These groups have been studied for decades in classical homotopy theory, and remain a vibrant research problem with no end in sight. (Contrary to one's first guess, they can be nontrivial even when $k > n$, such as $\pi_3(S^2) \cong \mathbb{Z}$ and $\pi_4(S^3) \cong \mathbb{Z}/2\mathbb{Z}$.) The appearance of these numerical invariants so close to the foundations of HoTT/UF suggests that like HoTT/UF, these invariants arise unavoidably from the Leibnizian concept of equality.

[16]Of course, one can define the formal system of HoTT/UF *inside* ZFC/FOL and then prove meta-statements in the latter about what can be proven by the former, but this is a transparent dodge. For all intents and purposes, synthetic homotopy theory is unique to HoTT/UF.

[17]And conversely, if we choose HoTT/UF as "the" foundation of mathematics, we can regard ZFC/FOL as a DSL for reasoning about well-founded hierarchies.

6. Synthetic and modal mathematics

As intriguing as this is, it only scratches the surface of the possibilities of synthetic mathematics. For if the primitive objects of mathematics can behave like either sets *or* ∞-groupoids, it stands to reason they can behave like many other things as well. And indeed, fields such as synthetic topology, synthetic differential geometry, synthetic domain theory, and so on, promise foundations for mathematics in which other "structures" are automatically present on all the primitive objects.

For example, in *synthetic topology*, the primitive objects are a sort of "topological space", all constructed objects automatically have an appropriate "topology", and all functions are "continuous". I put these words in quotes because the relationship of the synthetic concepts to their namesakes in ZFC/FOL is, again, one of analogy and interpretation. Rather than continuity being a defined *property* of a function, in synthetic topology we don't even need to talk about continuity because the theory prevents us from defining anything that isn't continuous. Like isomorphism-invariance in HoTT/UF, this is arguably a better fit for informal mathematics, where objects usually inherit topologies in a canonical way from their constructions, and most naturally defined functions are continuous, but we don't usually bother to check continuity formally. For example, "completions" of algebraic structures such as groups and rings must often be considered not as discrete structures but as *topological* ones; if we work in synthetic topology, such completion topologies appear automatically without requiring any extra work to construct, and cannot be accidentally forgotten. And also analogously to HoTT/UF (the "synthetic theory of ∞-groupoids"), synthetic topology has models using topological spaces in ZFC/FOL, so that synthetic proofs automatically yield "analytic" ones.

On the other hand, sometimes we do want to discuss discontinuous functions. We make this possible in synthetic topology with a *modality*: for every "space" A there is another space $\flat A$ that represents "A retopologized discretely". A "discontinuous function $A \to B$" is then defined as a (continuous) function $\flat A \to B$.

There are many other kinds of synthetic mathematics, each with their own modalities. Synthetic topology also admits a modality $\sharp$, which retopologizes a space indiscretely, and $\int$, which computes the homotopy type of a space. The latter connects topological spaces with their homotopy types; in [Shu18] I used it to prove a version of Brouwer's fixed-point theorem, which uses homotopy-theoretic machinery to construrct a strict fixed point.

Synthetic differential geometry admits these same modalities and also a triple $\Re$, $\Im$, and & that manipulate the "infinitesimal directions" of a smooth space. *Synthetic guarded domain theory* admits modalities $\triangleright$ for a recursive function call that "happens later" and $\square$ for something "at all times". And in *synthetic category theory* the primitive objects behave like ∞-*categories*, with modalities for opposite categories and maximal subgroupoids. Each of these theories has models in certain (∞-)toposes in ZFC/FOL, but goes beyond these models to establish a new kind of mathematics. And the possibilities for other such theories are endless by applying recent work on general *modal type theories* [GKNB21, GCK$^+$22, Shu23].

However, reasoning informally in modal type theories is difficult, because their rules tend to modify the context in novel ways. Here the *context* refers to the assumptions and variables that are available at some point during a definition or proof. For instance, if we prove the infinitude of primes by assuming for contradiction that there are only finitely many $p_1, \dots, p_n$, then the variables n and $p_1, \dots, p_n$ are placed into the context, along with the assumption that each p_i is prime.

Ordinarily, we expect that once a variable or assumption is placed in the context, it remains usable for the rest of the proof (or case or sub-proof). But modal type theories break this expectation. For example, if defining a function $f : A \to \flat B$, we might start by writing $f(x) = \dots$, with x an element of A now in the context of the right-hand side of the equality. But since A is not a $\flat$ type, the rule for constructing elements of $\flat B$ then *removes* this variable from the context, so that we cannot actually use it in defining the value of $f(x)$. Thus, absent any other information, the only functions $f : A \to \flat B$ we can define are constant ones (which makes sense topologically if $\flat B$ is discrete, since as far as we know A might be connected).

This is tricky even with one modality, and the problem grows nonlinearly as more modalities are added with rules governing their interactions. In [Shu18] I wrote informal mathematics in synthetic topology with $\flat$, $\sharp$, and $\int$, but this was already difficult, even though I used a theory in which only $\flat$ and $\sharp$ modify the context. Fortunately, proof assistants are already designed to manage contexts interactively: at each step the proof assistant displays all the variables and assumptions in the context. Thus, proof assistants for general modal type theories (currently under development [SGB22]) should make these theories more practically usable.

Of course, with present-day technology, such proof assistants will also reach a limited audience due to their inherent difficulty. But when enhanced with autoformalization, they have the potential to make synthetic mathematics more widely accessible, opening a gateway to many entirely new kinds of mathematics.

7. Conclusion

So how will machines change mathematics? Just as they always have, only more so: by amplifying the ability of human mathematicians to peer into the unknown. Not only by allowing lone mathematicians to create immense proofs that previously not even large teams could handle, but by making ever more complicated realms of mathematics practical to create and explore.

I have discussed homotopy type theory and modal type theories in this connection because they are an area of current research that I am familiar with, and in which it seems clear that advances in proof assistant technology have the potential to make a complicated system much more tractable. However, I suspect that they are only a fraction of the "strange new universes" waiting to be created, with powerful and flexible proof assistants to help us explore them.

References

[AGG$^+$22] Ayush Agrawal, Siddhartha Gadgil, Navin Goyal, Ashvni Narayanan, and Anand Tadipatri, *Towards a mathematics formalisation assistant using large language models*, `arXiv:2211.07524`, 2022.

[Bau20] Andrej Bauer, *What makes dependent type theory more suitable than set theory for proof assistants?* MathOverflow, 2020. `https://mathoverflow.net/q/376973` (version: 2020-12-07).

[FFLL16] Kuen-Bang Hou, Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine, *A mechanization of the Blakers-Massey connectivity theorem in homotopy type theory*, Proceedings of the 31st Annual ACM-IEEE Symposium on Logic in Computer Science (LICS 2016), ACM, New York, 2016, pp. 10, DOI 10.1145/2933575.2934545. MR3776776

[GCK$^+$22] Daniel Gratzer, Evan Cavallo, G. A. Kavvos, Adrien Guatto, and Lars Birkedal, *Modalities and parametric adjoints*, ACM Trans. Comput. Log. **23** (2022), no. 3, Art. 18, 29, DOI 10.1145/3514241. MR4461936

[GKNB21] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal, *Multimodal dependent type theory*, Log. Methods Comput. Sci. **17** (2021), no. 3, Paper No. 11, 67, DOI 10.46298/lmcs-17(3:11)2021. MR4298414

[Gon08] Georges Gonthier, *Formal proof—the four-color theorem*, Notices Amer. Math. Soc. **55** (2008), no. 11, 1382–1393. MR2463991

[HAB$^+$17] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Jason Rute, Alexey Solovyev, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller, *A formal proof of the Kepler conjecture*, Forum Math. Pi **5** (2017), e2, 29, DOI 10.1017/fmp.2017.1. MR3659768

[JSS$^+$22] Albert Jiang, Charles Edgar Staats, Christian Szegedy, Markus Rabe, Mateja Jamnik, Wenda Li, and Yuhuai Tony Wu, *Autoformalization with large language models*, In NeurIPS, 2022. To appear. `arXiv:2205.12615`.

[JWZ$^+$23] Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample, *Draft, sketch, and prove: Guiding formal theorem provers with informal proofs*, In ICLR, 2023. `arXiv:2210.12283`.

[KL21] Krzysztof Kapulkin and Peter LeFanu Lumsdaine, *The simplicial model of univalent foundations (after Voevodsky)*, J. Eur. Math. Soc. (JEMS) **23** (2021), no. 6, 2071–2126, DOI 10.4171/JEMS/1050. MR4244523

[Law05] F. William Lawvere, *An elementary theory of the category of sets (long version) with commentary*, Repr. Theory Appl. Categ. **11** (2005), 1–35. Reprinted and expanded from Proc. Nat. Acad. Sci. U.S.A. **52** (1964) [MR0172807]; With comments by the author and Colin McLarty. MR2177727

[LS13] Daniel R. Licata and Michael Shulman, *Calculating the fundamental group of the circle in homotopy type theory*, 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2013), IEEE Computer Soc., Los Alamitos, CA, 2013, pp. 223–232, DOI 10.1109/LICS.2013.28. MR3323808

[Lur09] Jacob Lurie, *Higher topos theory*, Annals of Mathematics Studies, vol. 170, Princeton University Press, Princeton, NJ, 2009, DOI 10.1515/9781400830558. MR2522659

[Lur14] Jacob Lurie, *Higher algebra*, Available at `http://www.math.harvard.edu/~lurie/`, Sep 2014.

[ML75] Per Martin-Löf, *An intuitionistic theory of types: predicative part*, Logic Colloquium '73 (Bristol, 1973), Stud. Logic Found. Math., Vol. 80, North-Holland, Amsterdam-Oxford, 1975, pp. 73–118. MR387009

[ML84] Per Martin-Löf, *Intuitionistic type theory*, Studies in Proof Theory. Lecture Notes, vol. 1, Bibliopolis, Naples, 1984. Notes by Giovanni Sambin. MR769301

[Sch21] Peter Scholze, *Half a year of the liquid tensor experiment: Amazing developments*, `https://xenaproject.wordpress.com/2021/06/05/half-a-year-of-the-liquid-tensor-experiment-amazing-developments/`, June 2021.

[SGB22] Philipp Stassen, Daniel Gratzer, and Lars Birkedal, `mitten`*: a flexible multimodal proof assistant*, 28th International Conference on Types for Proofs and Programs, LIPIcs. Leibniz Int. Proc. Inform., vol. 269, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2023, pp. Paper No. 6, 23, DOI 10.4230/lipics.types.2022.6. MR4627539

[Shu11] Michael Shulman, *A formal proof that* $\pi_1(s^1) = \mathbb{Z}$, `http://homotopytypetheory.org/2011/04/29/a-formal-proof-that-pi1s1-is-z/`, 2011.

[Shu16] Michael Shulman, *Idempotents in intensional type theory*, Log. Methods Comput. Sci. **12** (2016), no. 3, Paper No. 10, 24, DOI 10.2168/lmcs-12(3:9)2016. MR3548859

[Shu18] Michael Shulman, *Brouwer's fixed-point theorem in real-cohesive homotopy type theory*, Math. Structures Comput. Sci. **28** (2018), no. 6, 856–941, DOI 10.1017/S0960129517000147. MR3798599

[Shu19] Michael Shulman, *All* $(\infty, 1)$*-toposes have strict univalent universes*, `arXiv:1904.07004`, 2019.

[Shu20] Michael Shulman, *Why doesn't mathematics collapse even though humans quite often make mistakes in their proofs?* MathOverflow, 2020. `https://mathoverflow.net/q/338620` (version: 2020-06-15).

[Shu23] Michael Shulman, *Semantics of multimodal adjoint type theory*, Electronic Notes in Theoretical Informatics and Computer Science, vol. **3** Proceeding of MFPS XXXIX, `arXiv:2303.02572`, (2023).

[Uni13] The Univalent Foundations Program, *Homotopy type theory—univalent foundations of mathematics*, The Univalent Foundations Program, Princeton, NJ; Institute for Advanced Study (IAS), Princeton, NJ, 2013. MR3204653

Department of Mathematics, University of San Diego, San Diego, California 92110
Email address: `shulman@sandiego.edu`