

AUTOFORMALIZING THE CALCULATION OF $\pi_3(S^2)$

DANIEL CARRANZA, CHUNYI LIU, EMILY RIEHL, AND EGBERT RIJKE

ABSTRACT. We report our findings and the results of our experiment of autoformalizing the homotopy type theoretic computation of $\pi_3(S^2)$ in agda-unimath using Codex.

1 INTRODUCTION

In April 2026, the Lean Focused Research Organization released a Lean AI formalization benchmark lean-eval [Lea]. The purpose of lean-eval is to evaluate AI-powered autoformalization systems that are designed to attempt computer formalizations of challenging mathematical problems. The benchmark problems were chosen so as to have Lean-formalizable theorem statements using Mathlib, Lean’s mathematics library [Mat20], with known natural language proofs that would take significant work to formalize.

One of the benchmark problems was to demonstrate an isomorphism $\pi_3(S^2) \cong \mathbb{Z}$, calculating the third homotopy group of the 2-sphere [Mor]. This calculation involves some beautiful mathematical ideas: the generating element in the group $\mathbb{Z}$ corresponds under the isomorphism to a continuous map $S^3 \rightarrow S^2$ known as the *Hopf fibration*, a fiber bundle with fiber S^1 (see [Joh] for a visualization). This problem attracted our interest as a potential test of autoformalization capabilities in the proof assistant Agda using the agda-unimath library, a library of mathematical statements proven in homotopy type theory [RSPB+21].

While Lean is the most popular interactive theorem prover used by mathematicians today, it is not the only option, and historic successes have been achieved with many proof assistants. Perhaps the first great achievement in formalized mathematics was the verification of Landau’s *Grundlagen* in Automath [Ben79]. Other notable achievements include the formalization of the Jordan curve theorem in Mizar [Kor08], the formalizations of the four-color theorem and the Feit–Thompson theorem in Rocq (nee. Coq) [GAAB+13], and the formalization of the Kepler conjecture in HOL Light and Isabelle [HABD+17].

An interesting possibility enabled by the computer proof assistant paradigm, described in [Shu24], is to write proofs in a domain-specific language or “synthetic” mathematical framework. For instance, if one extends Martin-Löf’s dependent type theory with propositional truncations, function extensionality, uniqueness of identity proofs, and the axiom of choice, one gets the domain-specific language of sets, which is used in Mathlib. Similarly, *homotopy type theory* is the domain-specific language of homotopy types or spaces, in which Martin-Löf’s dependent type theory is extended with the univalence axiom and higher inductive types, and uniqueness of identity proofs

is dropped. Other domain-specific approaches to formalization include parametrized spectra and synthetic algebraic geometry. We will focus here on homotopy type theory.

The primitive objects in homotopy type theory—that is, the types—are homotopy types or spaces, and everything definable in homotopy type theory is automatically invariant under homotopy. Familiar spaces such as the spheres S^n are defined as higher inductive types, essentially as homotopy pushouts, rather than as subspaces of Euclidean spaces. For each pointed type A , we can define the n^{th} homotopy group $\pi_n(A)$, which is an example of a “homotopy invariant” construction, as the set-truncation of the space of pointed maps from S^n to A . Computations of homotopy groups within the setting of homotopy type theory have to be done entirely through homotopy-invariant methods.

In particular, there is a construction of an isomorphism $\pi_3(S^2) \cong \mathbb{Z}$ in homotopy type theory [Uni13, Corollary 8.6.19], which we will describe in more detail in §2. This proof mirrors in some ways and departs in others from the “analytic” proof in traditional foundations. An advantage of doing homotopy theory “synthetically” in this manner is that proofs and constructions apply to all models of homotopy type theory, whereas proofs in point-set topology only apply to point-set topology; in particular, all constructions in homotopy type theory apply to all Grothendieck ∞ -topoi [Shu19].

1.1 Open questions One motivation for domain-specific “synthetic” formal systems is their amenability to computer formalization [Voe14], and in fact the construction in homotopy type theory of the isomorphism $\pi_3(S^2) \cong \mathbb{Z}$ has been formalized multiple times in multiple proof assistants.¹ However, homotopy type theory libraries tend to be much smaller than Mathlib, providing less training data for an AI. Would an autoformalization agent, with no specific training in homotopy type theory, be able to formalize the homotopy type theory proof of this theorem?

We had a specific test library in mind, which like Mathlib had sufficient development to state the target theorem $\pi_3(S^2) \cong \mathbb{Z}$, but was missing the most significant ingredients in its proofs. The *agda-unimath* library was created in 2021 with the ambition to serve as a general purpose library for formalized mathematics in univalent foundations [RSPB+21].

One of our motivations for conducting this experiment is to test autoformalization capabilities in an area of synthetic mathematics. Another motivation is to attempt to address an equally important question of whether autoformalization of mathematics can be done in a responsible way that ultimately supports the libraries that it relies upon. In recent months, there have been occasions where autoformalizers have arguably hurt human-led formalization efforts more than helping them. In an incident described in [Avi26], a competing autoformalization project piggybacked upon and then stole credit from a human-led collaboration, without contributing much in the way of usable code. Arguably none of the AI-generated code was of “library quality,” suitable for being contributed back into Mathlib.

¹Among the currently actively libraries, see the [Cubical Agda formalization](#) and the [Coq-HoTT formalization](#). The latter isomorphism was actually added after our experiment was complete, though for the reasons noted below, all of the essential ingredients for this calculation were already present in that library.

1.2 Overview of results and outline The task was for Codex to successfully formalize the computation of $\pi_3(S^2)$, with human prompts for instructions but otherwise in an autonomous manner. Importantly, no human was to supply any piece of the code. Secondary, we set the goal of having it produce high-quality code suitable for the agda-unimath library. In particular, our aim was to autoformalize intermediate results in full and not just the portions of those theorems needed for the isomorphism $\pi_3(S^2) \cong \mathbb{Z}$.

Initially, we provided Codex with a project description, explaining the overall goal of the project and the methods we expected it to be using. Based on this description, we used an autogenerated formalization plan, with intermediate targets. After a little under one month (June 3 to June 25, 2026), Codex, guided by the plan and our repeated prompting, was able to complete the formalization plan, producing a formal proof of $\pi_3(S^2) \cong \mathbb{Z}$ in homotopy type theory.

All of our code can be found on github, which also contains anonymized versions of our full codex session logs, including our prompts and the model’s responses:

github.com/emilyriehl/Codex-Homotopy-Group/tree/arxiv-v1

In §2, we give a brief introduction to homotopy type theory and then sketch the proof of the calculation $\pi_3(S^2) \cong \mathbb{Z}$ in homotopy type theory following [Uni13]. We also give more background about the agda-unimath library.

In §3, we give more details about our experimental setup, our resource usage, and the results, in particular describing the other formalization milestones achieved along the way. We also highlight bottlenecks and other inefficiencies and describe how they were overcome.

In §4, we record our takeaways from the experiment, list various future technical improvements, and revisit the important discussion of what this autoformalization means for the agda-unimath library.

1.3 Acknowledgments The first, third, and fourth authors are supported by a grant through DARPA’s Exponentiating Mathematics program (HR0011262E019). The credits used for this autoformalization experiment were donated by OpenAI through this initiative.

In addition, the first author is also supported by the NSERC, while the third author is also supported by the NSF (DMS-2507077) and by the AFOSR (FA9550261B013). Help setting up Codex was provided by the Johns Hopkins Data Science and AI Institute, which employs the second author.

While AI was used, with human supervision, to generate a plan for the experiment, generate the experimental logs, and generate the Agda code and documentation, this paper describing our results was entirely written by the four human authors listed above.

As described in the text below, our Codex autoformalization succeeded because it was able to learn from the human authors of the original homotopy type theory proofs as well as past work by human experts formalizing these results in other homotopy type theory libraries. We have endeavored to provide precise references throughout the text and wish to convey our appreciation for this beautiful mathematics.

2 MATHEMATICAL TARGET AND CONTEXT

2.1 Basic concepts of homotopy type theory In homotopy type theory, the axiom *uniqueness of identity proofs* is dropped in favor of the *homotopy interpretation of type theory*, where types are spaces², dependent types are fibrations, and the type $a = b$ of identifications between two points a and b of the same type is the type of paths between them [AW09]. The type $a = b$ exists whenever a and b are points belonging to the same type, but it is not necessarily the case that $a = b$ itself contains any points. We write $p : a = b$ to indicate that p is a path from a to b . The path p is a witness that a and b are “equal,” identifying a with b in an explicit structural sense, as is consistent with the common mathematical practice of treating two objects as the same if they are the same in a structural sense. In particular, for any a , we have $\text{refl} : a = a$, and moreover the family of identity types associated to a type A is freely generated by these “constant” paths: it is an inductive type with refl as its only constructor.

Given two identifications $p, q : a = b$, we might ask whether they are themselves the same. The type-theoretic way of stating that p and q are the same is via the identity type $p = q$, and we see here at once that this process can be iterated, giving rise to a tower of iterated identity types. In particular, we can define the *iterated loop spaces* of a pointed type (A, a) by

$$\Omega^0(A, a) := (A, a) \quad \text{and} \quad \Omega^{n+1}(A, a) := \Omega^n(a = a, \text{refl}).$$

When encountering a type in the context of homotopy type theory, one typically wants to characterize its identity type at the earliest occasion. The identity type of the universe is by the *univalence axiom* postulated to be equivalent to equivalences; that is, an identification of types is equivalent to an equivalence between them. This axiom implies that two sets are the same if there is a bijection between them, two algebraic structures such as two groups or two rings are the same if they are isomorphic, two subsets are the same if they contain the same elements, two equivalence relations are the same if they relate the same elements, and so on.

Identity types can also be used in the specification of a new kind of inductive type: *higher inductive types*. The first example of a higher inductive type is the circle, which is a type S^1 generated by a point pt and a loop

$$\text{loop} : \text{pt} = \text{pt}.$$

The circle is then equipped with an induction principle, which tells us that in order to construct a section of a fibration over S^1 , we have to construct the value of this section at pt and show that it is a fixed point (as stated using identity types) under the transport of loop . A special case of this induction principle is the universal property of the circle, which states that a map $S^1 \rightarrow X$ is equivalently described as a free loop (x, ω) in X , where $x : X$ and $\omega : x = x$. That is, we have an equivalence

$$(S^1 \rightarrow X) \simeq \sum_{x : X} x = x.$$

²More precisely, the types in homotopy type theory should be thought of as homotopy types, or ∞ -groupoids, or anima—or, more generally, as objects in an ∞ -topos. We adopt the common homotopy theorists’ convention of using “spaces” for these notions.

A beautiful argument via the encode-decode method shows how the univalence axiom can be used to obtain an equivalence, and indeed a group isomorphism, $\Omega S^1 \simeq \mathbb{Z}$ [LS13].

More generally, we can define the *suspension* of a type as a higher inductive type ΣX with two points $n, s : \Sigma X$ and a map

$$\text{merid} : X \rightarrow n = s.$$

The universal property of the suspension states that the evaluation map

$$(\Sigma X \rightarrow Y) \rightarrow \left(\sum_{y,z : Y} X \rightarrow y = z \right),$$

given by $f \mapsto (f(n), f(s), \text{ap}_f \circ \text{merid})$ is an equivalence. Here, the function

$$\text{ap}_f : x = y \rightarrow f(x) = f(y),$$

called the *action on identifications* of f , is defined inductively by $\text{ap}_f(\text{refl}) := \text{refl}$.

Given pointed types A and B , with implicit basepoints, we can reformulate the universal property of the suspension as the familiar adjunction

$$(\Sigma A \rightarrow_* B) \simeq_* (A \rightarrow_* \Omega B).$$

Here, we use $\rightarrow_*$ to denote the type of pointed maps, and indeed the equivalence of the pointed universal property of suspension is a pointed equivalence.

By iterating the suspension, we can define the n -sphere for each n :

$$S^{n+1} := \Sigma^n S^1.$$

The universal property of the n -sphere S^n provides an equivalence

$$(S^n \rightarrow_* B) \simeq_* \Omega^n B$$

for any pointed type B .

We note that a type X is considered to be a *set*, or homotopically discrete, if every map $S^1 \rightarrow X$ is constant, in the sense that the diagonal map

$$X \rightarrow (S^1 \rightarrow X)$$

given by $x \mapsto (t \mapsto x)$ is an equivalence. In particular, in homotopy type theory we classify only some types as sets. This way, we view homotopy type theory as an *extension* of traditional set-level mathematics.

Using the idea of higher inductive types, we can also define the *set truncation* of a type X . The set truncation of X is a type $\|X\|_0$, which is a set according to the previous definition, that comes equipped with a map $X \rightarrow \|X\|_0$ that is universal among maps from X to a set. There are many equivalent ways of constructing set truncations, and indeed it is customary to work only with its *specification* once it has been implemented. A fruitful way of thinking about the type $\|X\|_0$ is that it is the type of connected components of X .

More generally, we say that a type X is k -truncated if every map $S^{k+1} \rightarrow X$ is constant in the sense that the diagonal map

$$X \rightarrow (S^{k+1} \rightarrow X)$$

is an equivalence. Taking $S^0 := \{\pm 1\}$ and $S^{-1} := \emptyset$, this definition makes sense for $k \geq -2$. A (-2) -truncated type is called *contractible*, and a (-1) -truncated type is called a *mere proposition* [Uni13, Definition 3.3.1] or a *proposition* [Rij25, Definition 12.1.1]. The k -truncations $\|X\|_k$ can be defined for all $k \geq -2$, as the universal k -truncated type equipped with a map $X \rightarrow \|X\|_k$. A type X is k -connected if its k -truncation $\|X\|_k$ is contractible, while a map $f : X \rightarrow Y$ is k -connected if its fibers $\sum_{x:X} f(x) = y$ over all points $y : Y$ are k -connected. For example, the $(k+1)$ -sphere is k -connected.

The homotopy groups of pointed types A are defined in homotopy type theory as the set truncations of the iterated loop spaces:

$$\pi_n(A) := \|\Omega^n(A)\|_0.$$

For $n \geq 1$, the group structure on these sets is inherited from the groupoidal structure of the identity type, and the Eckmann–Hilton argument, showing that the homotopy groups are abelian for $n \geq 2$, is given in homotopy type theory via path induction [Uni13, Theorem 2.1.6]. A type X is k -connected if and only if its homotopy groups $\pi_n(X, x)$ are trivial for each $n \leq k$ and each $x : X$. For $k \geq 0$, a map $f : X \rightarrow Y$ is k -connected if and only if the induced map $\|X\|_0 \rightarrow \|Y\|_0$ is an equivalence and, for each $x : X$, the induced map on homotopy groups $\pi_n(X, x) \rightarrow \pi_n(Y, f(x))$ is an isomorphism for $n \leq k$ and is surjective when $n = k+1$ [Uni13, Corollary 8.8.5].

An important difference between homotopy type theory and point-set topology is that the condition that a type X is k -truncated is stronger than the condition that $\pi_n(X, x)$ is trivial for all $n > k$ and all $x : X$. There are models of homotopy type theory in which there is a map that is k -connected for all k , while not being an equivalence [Lur03, Counterexample 2.10.1 and Remark 2.10.2; Uni13, §8.8; Shu19].

2.2 The calculation of $\pi_3(S^2)$ in homotopy type theory The calculation of the group $\pi_3(S^2)$ in homotopy type theory has three major ingredients that we explain below:

- (i) The construction of the long exact sequence of homotopy groups of a fiber sequence.
- (ii) The construction of a particular fiber sequence $S^1 \rightarrow_* S^3 \rightarrow_* S^2$, called the Hopf fibration, which together with the long exact sequence gives isomorphisms $\pi_2(S^2) \cong \pi_1(S^1)$ and $\pi_3(S^3) \cong \pi_3(S^2)$.
- (iii) The Blakers–Massey theorem, which implies the Freudenthal suspension theorem, which gives an isomorphism $\pi_2(S^2) \cong \pi_3(S^3)$.

A *fiber sequence* $F \rightarrow_* E \rightarrow_* B$ in homotopy type theory consists of pointed maps $i : F \rightarrow_* E$ and $p : E \rightarrow_* B$ and a pointed homotopy H , which is often left implicit, witnessing that the following square commutes and is a homotopy pullback:

$$\begin{array}{ccc} F & \xrightarrow{i} & E \\ \downarrow & & \downarrow p \\ \mathbf{1} & \longrightarrow & B. \end{array}$$

Every fiber sequence is uniquely determined by a pointed map p of type $E \rightarrow_* B$, as the fiber, defined above, can be shown to define the homotopy pullback. However, the type of such data can be shown to be equivalent to the type that is given by a pair of pointed

maps, together with an equivalence between F and the fiber of p commuting with the inclusions into E , and this expanded form of the data is more convenient in situations, such as ours, where we wish to explicitly specify F , E , and B . The *agda-unimath* library is very deliberate about representation independent design choices such as these.

Any fiber sequence $F \rightarrow_* E \rightarrow_* B$ induces a fiber sequence $\Omega B \rightarrow_* F \rightarrow_* E$. Iterating this process, we obtain a tower of types $\cdots \rightarrow_* \Omega B \rightarrow_* F \rightarrow_* E \rightarrow_* B$. This construction was first formalized by Voevodsky in [Voe10]. Applying the set truncation to this tower yields the *long exact sequence* of homotopy groups [Uni13, Theorem 8.4.6; AKL15, Theorem 4.3.4]:

$$\cdots \longrightarrow \pi_{n+1}(B) \xrightarrow{\partial} \pi_n(F) \xrightarrow{i_*} \pi_n(E) \xrightarrow{p_*} \pi_n(B) \xrightarrow{\partial} \pi_{n-1}(F) \longrightarrow \cdots.$$

The Hopf fibration $S^1 \rightarrow_* S^3 \rightarrow_* S^2$ is a fiber sequence constructed as a special case of the *Hopf construction* associated to a connected H -space A . An H -space is a type A equipped with a binary operation $\mu : A \times A \rightarrow A$ and a unit $e : A$ equipped with identifications $\lambda_x : \mu(e, x) = x$ and $\rho_x : \mu(x, e) = x$ for each $x : A$ [Uni13, Definition 8.5.4]. In the context of homotopy type theory, it is often beneficial to assume an extra coherence condition: an identification $\lambda_e = \rho_e$ [BR18]. Such H -spaces are called *coherent* [BCFR25]. The associated fiber sequence $A \rightarrow_* A * A \rightarrow_* \Sigma A$ has fiber A , base space the suspension of A , and total space $A * A$ [Uni13, Lemma 8.5.7], where the *join* of types A and B is defined to be the homotopy pushout:

$$\begin{array}{ccc} A \times B & \xrightarrow{\pi_B} & B \\ \pi_A \downarrow & & \downarrow \\ A & \longrightarrow & A * B. \end{array}$$

The classical Hopf fibration $S^1 \rightarrow_* S^3 \rightarrow_* S^2$ arises when this construction is specialized to the H -space S^1 . Here, the H -space structure is given by a multiplication map $S^1 \times S^1 \rightarrow S^1$ corresponding to complex multiplication on the unit circle. Such a map can indeed be defined synthetically via the universal property and induction principle of the circle [Uni13, Lemma 8.5.8]. By the Hopf construction, we obtain a fiber sequence $S^1 \rightarrow_* S^1 * S^1 \rightarrow_* S^2$. Using the equivalence $S^1 * S^1 \simeq S^3$, which requires a proof, we obtain the Hopf fibration. Having constructed the Hopf fibration on the 2-sphere, we obtain a long exact sequence:

$$\cdots \longrightarrow \pi_{n+1}(S^2) \xrightarrow{\partial} \pi_n(S^1) \xrightarrow{i_*} \pi_n(S^3) \xrightarrow{p_*} \pi_n(S^2) \xrightarrow{\partial} \pi_{n-1}(S^1) \longrightarrow \cdots.$$

Since the circle is 1-truncated, we have $\pi_n(S^1) \cong 0$ for $n > 1$. Furthermore, since S^2 is 1-connected and S^3 is 2-connected, we have that $\pi_1(S^2) \cong 0$ and $\pi_2(S^3) \cong 0$. Thus, we obtain the following exact sequence of groups:

$$0 \longrightarrow \pi_3(S^3) \longrightarrow \pi_3(S^2) \longrightarrow 0 \longrightarrow 0 \longrightarrow \pi_2(S^2) \longrightarrow \pi_1(S^1) \longrightarrow 0,$$

giving isomorphisms $\pi_3(S^3) \cong \pi_3(S^2)$ and $\pi_2(S^2) \cong \pi_1(S^1)$. In particular, since $\pi_1(S^1) \cong \mathbb{Z}$ was proven in homotopy type theory in [LS13], we find that $\pi_2(S^2) \cong \mathbb{Z}$.

The computation of $\pi_3(S^2)$ therefore requires one more ingredient: a proof that $\pi_2(S^2) \cong \pi_3(S^3)$. This isomorphism is obtained from the Freudenthal suspension theorem, which asserts that for any pointed n -connected type A for $n \geq -1$, the map

$$A \rightarrow \Omega \Sigma A$$

is $2n$ -connected. LeFanu Lumsdaine’s beautiful proof was reported in [Uni13, Theorem 8.6.4] and a formalization was reported in [Lic13]. Since the 2-sphere is 1-connected, we see that the map

$$S^2 \rightarrow \Omega S^3$$

is 2-connected, so that it induces an isomorphism $\pi_2(S^2) \cong \pi_3(S^3)$. Thus, the computation $\pi_3(S^2) \cong \mathbb{Z}$ in homotopy type theory is completed.

The Freudenthal suspension theorem in homotopy type theory is typically derived as a corollary of the Blakers–Massey theorem. The Blakers–Massey theorem asserts that for a homotopy pushout square

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ g \downarrow & & \downarrow \\ C & \longrightarrow & D, \end{array}$$

if the map $f : A \rightarrow B$ is m -connected and the map $g : A \rightarrow C$ is n -connected, then the *gap map*, which is the universal map into the homotopy pullback $A \rightarrow B \times_D C$, is $(m + n)$ -connected. To obtain the Freudenthal suspension theorem from Blakers–Massey, take $B := \mathbf{1}$ and $C := \mathbf{1}$, so that the homotopy pushout D is the suspension ΣA and the homotopy pullback is $\Omega \Sigma A$. The Blakers–Massey theorem was first proven and formalized in homotopy type theory in [HFL16].

2.3 The agda-unimath library The agda-unimath library is one of the largest formalization projects for the Agda proof assistant. It currently contains over 3000 pages, with concepts and theorems from a wide variety of mathematical subjects. It is built on the univalent foundations for mathematics, and treats all its subjects from a univalent point of view. Although its foundations are homotopy type theory, synthetic homotopy theory is not the main focus of the library, which made it suitable for our experiment.

The agda-unimath library is organized by mathematical subject. The foundational topics of univalent mathematics are treated in the folders `foundation` and `foundation-core`. These are organized in two folders for bootstrapping purposes, so as to avoid cyclic dependencies of the modules. The central folders for our experiment are:

```
foundation
foundation-core
group-theory
structured-types
synthetic-homotopy-theory
```

Within the `foundation` folder, one finds modules such as `identity-types`, `equivalences`, and `truncated-types`. The library is designed around a *one-concept-per-file* paradigm. The organization of each file is modeled after *nLab* pages: Modules start with an *Idea*-section, explaining in natural language the idea of a concept or named entity, then proceeds with one or several definitions of that concept, and afterwards proves basic properties about it. References and pointers to related files may be given

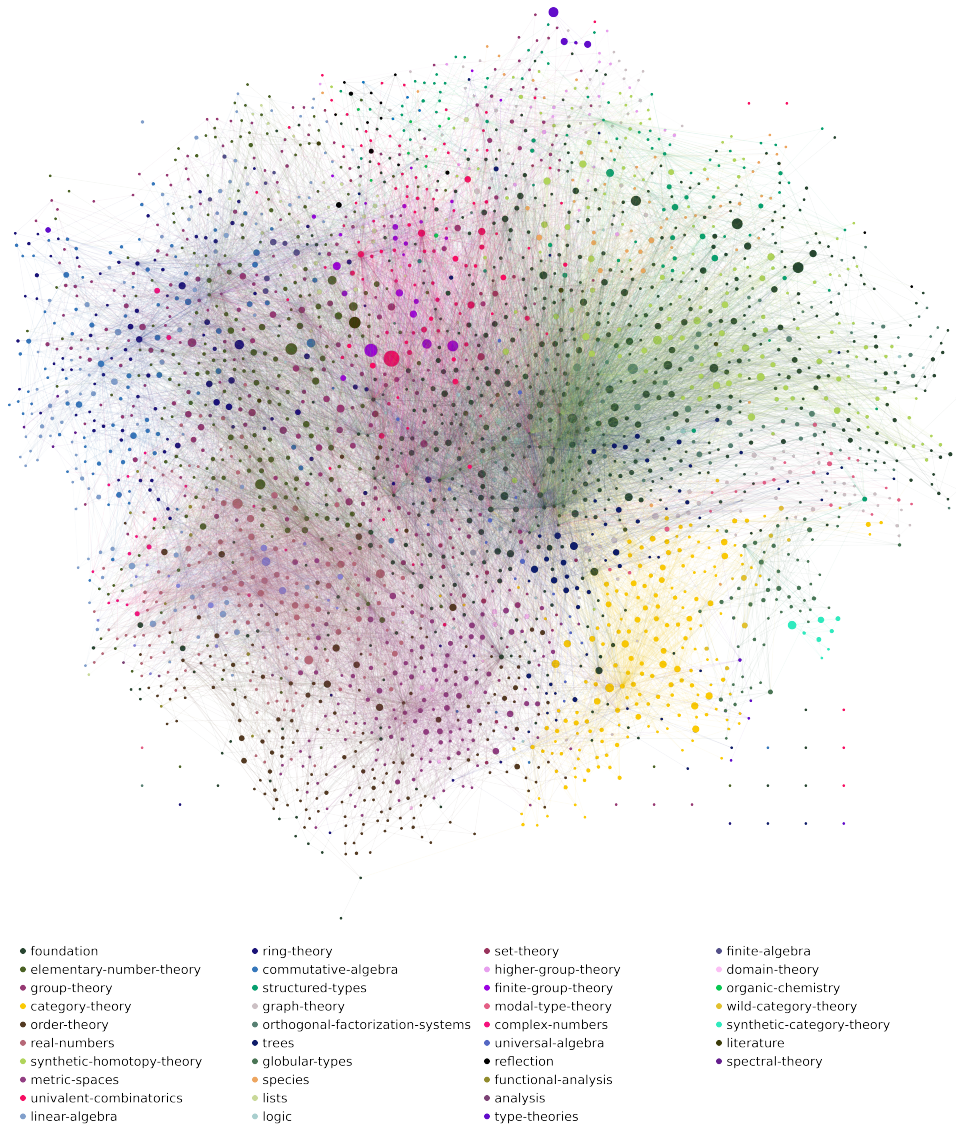


Figure 1: Dependency graph of the agda-unimath library, created by Fredrik Bakke.

at the end of the module. The agda-unimath library uses sharp, disambiguative conventions for concept declarations, where everything that can be named or disambiguated in natural language gets its own file. As an example, we show an excerpt from the module `synthetic-homotopy-theory.circle`, in which the standard circle is postulated.

```
## Idea
```



```

**The circle** is the initial type equipped with a base point and a
[loop](synthetic-homotopy-theory.loop-spaces.md).

## Postulates

```agda
postulate
 S1 : UU lzero

postulate
 base-S1 : S1

postulate
 loop-S1 : base-S1 = base-S1

free-loop-S1 : free-loop S1
free-loop-S1 = base-S1 , loop-S1

S1-Pointed-Type : Pointed-Type lzero
S1-Pointed-Type = S1 , base-S1

postulate
 ind-S1 : induction-principle-circle free-loop-S1
```

```

Here, the entry `induction-principle-circle` is defined in the module `synthetic-homotopy-theory.universal-property-circle` which specifies what a circle is in homotopy type theory without declaring the standard circle. The infrastructure written for circles is written representation independently so as to be applicable to any type that is a circle, for instance, the type $S^0 * S^0$.

Finally, we note that `agda-unimath` formalizes the theory of groups in two parallel ways. The ordinary notion of group is the usual one, where a group is a set with a binary operation and a unit element, satisfying the axioms of a group. We also formalize the notion of *concrete group*, which is a pointed connected 1-type. The idea is that identifications of an element with itself can be thought of as the *symmetries* of that element. For example, the type of n -element types is the concrete group presenting the n^{th} symmetric group S_n , since the identifications of the standard n -element type Fin_n are bijections $\text{Fin}_n \cong \text{Fin}_n$. Thus, a concrete group *concretely* presents a group as the group of symmetries of an object. Given a pointed connected 1-type BG , the underlying type of the group is $G := \Omega BG$. From the perspective of concrete groups, a group homomorphism $G \rightarrow H$ is a pointed map $BG \rightarrow_* BH$. All of group theory can be developed in this way [BBCDG26]. The idea of treating group theory via classifying spaces in the context of homotopy type theory was first explored in [Shu15] and a first systematic study was reported in [BDR18].

At the moment of our experiment, `agda-unimath` defined the *concrete* homotopy groups as follows:

```

    ### The concrete homotopy groups

    ```agda
 module _
 {l : Level} (n : ℕ) (A : Pointed-Type l)
 where

 concrete-homotopy-group : Concrete-Group l
 concrete-homotopy-group =
 concrete-group-Pointed-Type (iterated-loop-space n A)
    ```
    
```

Here, the concrete group associated to a pointed type (A, a) is the 1-truncation of the connected component of the base point a . Since the underlying set of a concrete group BG is its loop space, we see here that `concrete-homotopy-group n A` is the classifying space of $\pi_{n+1}(A)$. This unfortunate shift in indexing will come to play a part in our experiment.

3 DESIGN AND RESULTS OF THE EXPERIMENT

In this section, we describe the design and results of the autoformalization experiment, which was initiated on June 3rd, 2026; the main theorem was completed on June 25th, 2026.

3.1 Experimental design An ideal test of autoformalization capabilities in homotopy type theory involves a theorem that is stateable using the previously-defined infrastructure of an ambient library, but which requires a significant amount of work to prove. The calculation of $\pi_3(S^2) \cong \mathbb{Z}$ perfectly fit this desiderata. The `agda-unimath` library already contained definitions of the spheres and homotopy groups, the notion of a group isomorphism, and the group of integers. But it was missing most of the essential ingredients in this calculation: in particular the long exact sequence of a fibration, the Hopf fibration, and the Freudenthal suspension theorem.

After we identified our formalization target, we checked to see whether any human-led teams were working on any of these components. On a Discord server associated with the `agda-unimath` library, we wrote

One of the ways we're trying to be responsible in experimenting with AI for formalization is to check first whether any of our autoformalization targets are currently works in progress by humans who should be given priority. To that end is anyone currently working on formalizing the Hopf fibration or the calculation of $\pi_3(S^2)$ for agda-unimath? This also involves the long exact sequence of the fibration. ...right now we want to make sure that the mere existence of this code does not disrupt/conflict with/undermine any human efforts.

After we determined that there was no known formalization effort in this direction, we settled on this project.

As none of the first, third, and fourth authors had experience using a coding agent, we submitted a request for research software engineering support from the Johns Hopkins Data Science and AI Institute (DSAI), which put them in contact with the second author.

Using an email that the third author wrote to describe the project to colleagues at the DSAI as a prompt (preserved in the repository as `email.md`³), Claude Code generated a guiding formalization plan document⁴ that correctly identified a subtlety in the agda-unimath statement of the main goal:

Important indexing note: `synthetic-homotopy-theory.homotopy-groups` documents an “Obi-wan error” in `concrete-homotopy-group`: the index `n` names the $(n+1)^{\text{st}}$ abstract homotopy group. Therefore $\pi_3(S^2)$ should correspond to `concrete-homotopy-group 2 (sphere-Pointed-Type 2)`, not index 3. Candidate final shape:

```
iso-Group
( group-Concrete-Group
  ( concrete-homotopy-group 2 (sphere-Pointed-Type 2)))
( Z-Group)
```

A similar formalization plan produced by Codex failed to notice the off-by-one error, stating the theorem incorrectly as an isomorphism `iso-Concrete-Group (concrete-homotopy-group 3 sphere-2) integer-Concrete-Group`. Thus, we initiated Codex on the Claude Code-generated formalization plan.

We set up Codex in a shared private github repository so that the prompting could be done by any of the first, third, or fourth authors using multiple computers.

Before attempting any autoformalization, we provided Codex with skills and reference documentation, instructing it in the importance of using Agda `--without-K`, which is to say in homotopy type theory mode, and matching the agda-unimath [library design principles](#) and [style guides](#).

The skills are collections of instruction and reference documents that the agent is told to consult as it works, whose initial versions were generated using Claude Code from the agda-unimath source and documentation. The reference skill contains four documents describing:

- the writing conventions of the library;
- how to search the library’s more than 3000 files for an existing lemma;
- the foundational vocabulary of the library: universes, Σ -types, identity types, equivalences, and truncations;
- and how to interpret the Agda error messages that arise when working in this mode.

The workflow skill tells the agent how to work: search the library before proving a lemma from scratch, make small edits and typecheck often, never introduce a postulate, and keep a status report and a chat log up to date as a record of the experiment.

³github.com/emilyriehl/Codex-Homotopy-Group/blob/arxiv-v1/email.md

⁴github.com/emilyriehl/Codex-Homotopy-Group/blob/arxiv-v1/FORMALIZATION-PLAN.md

During the experiment, Codex extended this material at our prompting, for instance adding a fifth reference document on formalization practice in homotopy type theory⁵.

We also had to decide in advance how formalized code would be judged. Testing before the project began had shown that the Agda MCP server, an MCP server that gives coding agents interactive access to Agda (discussed further below), would report “ok-complete” for files it had merely scope-checked, passing code that Agda itself rejects. The repository therefore includes a short script, `check.sh`⁶, that runs the actual Agda binary on a file with the library flags applied and unsolved metavariables disallowed. The script requires no network access, so Codex could run it from inside its own sandbox. A module counted as proven only when `check.sh` ran cleanly and no holes remained in the file; the instructions to the agent state that nothing else would be considered success.

3.2 Initial autoformalization The initial formalization plan noted 35 dependencies, including the main theorem, 11 of which were missing:

- (i) the definition and basic theory of pointed fiber sequences,
- (ii) the long exact sequence of homotopy groups from a pointed fiber sequence,
- (iii) the isomorphism that arises from a four term exact sequence with zeros at its endpoints,
- (iv) the Hopf fibration $A \rightarrow_* A * A \rightarrow_* \Sigma A$ associated to a connected H -space A ,
- (v) the connected H -space structure on S^1 ,
- (vi) the Hopf fibration $S^1 \rightarrow_* S^3 \rightarrow_* S^2$ arising from equivalences $S^3 \simeq S^1 * S^1$ and $S^2 \simeq \Sigma S^1$,
- (vii) the computation $\pi_n S^1 \cong 0$ for $n > 1$,
- (viii) the Freudenthal suspension theorem as a corollary of the Blakers–Massey theorem (which was also missing),
- (ix) the isomorphisms $\pi_{n+3} S^{n+3} \cong \pi_{n+2} S^{n+2}$ for $n \geq 0$ that follow from Freudenthal, together with a separate base case⁷ $\pi_2 S^2 \cong \pi_1 S^1$,
- (x) the computation $\pi_n S^n \cong \mathbb{Z}$ for $n > 1$ (with the $n = 1$ case already in the library),
- (xi) the deduced isomorphism $\pi_3(S^2) \cong \mathbb{Z}$ arising from the exact sequence

$$0 \cong \pi_3(S^1) \rightarrow \pi_3(S^3) \rightarrow \pi_3(S^2) \rightarrow \pi_2 S^1 \cong 0.$$

The initial prompting on June 3rd targeting the first of these — the definition of general pointed fiber sequences — was done as a group with the code carefully reviewed. We requested a repackaging of the Codex-generated `fiber-sequence-Pointed-Type` including a renaming of some of its fields. We also directed Codex to use some existing API for the coherence involved in a commuting triangle of pointed maps.

⁵github.com/emilyriehl/Codex-Homotopy-Group/blob/arxiv-v1/.codex/skills/agda-unimath-reference/references/hott-skills.md

⁶github.com/emilyriehl/Codex-Homotopy-Group/blob/arxiv-v1/check.sh

⁷The formalization plan erroneously lists $\pi_3 S^3 \cong \pi_2 S^2$ as the required base case, but this is in the range of Freudenthal. The actually required base case is instead $\pi_2 S^2 \cong \pi_1 S^1$, which follows from the long exact sequence of the Hopf fibration as noted above. Codex autoformalized this by showing that $S^3 \simeq \Sigma \Sigma S^1$ is 2-connected, so its homotopy groups in degrees 1 and 2 vanish.

The next objective, to formalize the long exact sequence of homotopy groups associated to a fiber sequence of pointed types, was the first major test.

3.3 Progress reports Over the subsequent weeks, we made a number of errors associated to new users:

- After a few days, we learned how to change the model reasoning level to gpt-5.5 xhigh, which was used for the remainder of the experiment.
- After a few more days, we learned how to set up an API key associated with the organization which had the donated credits. This is why our project spending report dates from June 9th rather than the project start date of June 3rd.
- The following day, we realized that Codex was not configured to use the [agda-mcp-server](#) we had identified. Over the next few days, we figured out that this needed to be configured on each individual machine interacting with the git repo.

These last two improvements were made while the third author attending a workshop on “Formalization of Higher Categories” at the Mittag-Leffler institute and got some tips from colleagues. Andrej Bauer explained the use of plan mode to generate more complicated prompts and also suggested instructing Codex to attempt some top-down reasoning, decomposing the target isomorphism in terms of its easily-stable prerequisites, with holes left in the places where the formalized proofs could not be easily defined.

It is hard to say how much the MCP server helped, as we made no controlled comparison, and it was configured in the same week as the other corrections just described. We do know how much it was used, and how often it misled us. At the end of the experiment we asked Codex to extract every Agda MCP call from its own session logs. There were 711 of them, so the server was used constantly, mostly to load files, inspect goals, and infer types. But the extraction also flagged 290 instances of misleading behavior, including 70 replies that reported success while containing an Agda error, and a handful in which an editing tool claimed a goal was solved although the edited file no longer typechecked. The same warning recurs throughout the session logs: `check.sh`, and not the server, has the final word, because the server could report success on code that Agda rejects. The report we prepared for the server maintainers⁸ is discussed in §4.2.

3.4 Bottlenecks A major source of inefficiency was caused by the fact that the primary prompter, the third author, never learned how to reliably convince Codex to keep working without her direct supervision. Thus, very little was accomplished on days during which she was occupied by other tasks.⁹

There were various stages where the formalization targets seemed too hard for Codex without further expert guidance concerning how to handle the “invisible mathematics” that arises in computer formalization, which must concern itself with a level of detail that is absent in the natural-language literature. The most effective tool to overcome such instances involved pointing Codex to existing human formalizations

⁸github.com/emilyriehl/Codex-Homotopy-Group/tree/arxiv-v1/agda-mcp-ux-report

⁹After the completion of this experiment, the authors learned about goal mode.

found in the Coq-HoTT library [BGLS+17] using Rocq. Once Codex learned to use this library as a guide, it autoformalized a third hard theorem, the Blakers–Massey theorem, without our help.

The most significant difficulty arose at various points in the proof of exactness of the long exact sequence of homotopy groups defined from a fiber sequence of pointed types $F \rightarrow_* E \rightarrow_* B$. Without further guidance, Codex tended to pursue direct proofs of the next targets via messy path algebra that would ultimately fail, and then return unrelated incremental progress, leading the third author to prompt on July 16th:

We seem to get stuck at the same point. Can some of the other homotopy type theory repositories that have formalized the long exact sequence help provide inspiration?

Despite this generic guidance, Codex had difficulties formalizing the exactness of the higher stages of the long exact sequence, attempting to use multiple constructions of the map $\Omega^{n+1}B \rightarrow_* \Omega^n F$ and struggling to prove that they were homotopic, occasionally due to incompatibilities in sign conventions used either to define the iterated boundary maps or to identify the fibers of the iterated fiber sequences.

Success was finally achieved by more firmly instructing Codex to follow the approach of the Coq-HoTT library: from a general fiber sequence $F \rightarrow_* E \rightarrow_* B$, one can construct shifted fiber sequences $\Omega B \rightarrow_* F \rightarrow_* E$ and $\Omega E \rightarrow_* \Omega B \rightarrow_* F$. We instructed Codex to stay the course with a prompt on June 19th:

I'd prioritize the harder upfront equivalence. The goal is not just to have a proof but the right proof that we will ultimately upstream to the agda-unimath library. Please update your skills and general references to reflect this ultimate priority.

It took several days of repeated prompting to “work very hard” before this was completed on June 21st, at which point enough of the long exact sequence was in place to fit into the scaffold (though the results were not yet “library quality”).

On June 21st, Codex began work on the Hopf fibration. The agda-unimath library already had definitions of an H -space and a construction of the coherent multiplication operation on the circle, so Codex began by packaging this into an H -space structure on the circle and then defining the fiber sequence $A \rightarrow_* A * A \rightarrow_* \Sigma A$ associated to a general H -space A . As the spheres are defined in agda-unimath by iterated suspension, identifying the sequence $S^1 \rightarrow_* S^1 * S^1 \rightarrow_* \Sigma S^1$ with a fiber sequence $S^1 \rightarrow_* S^3 \rightarrow_* S^2$ only required an equivalence $S^1 * S^1 \simeq S^3$, which proved to be surprisingly difficult.

One way to understand this equivalence is to redefine the spheres as iterated join powers of the 2-point type Fin_2 . For this, Codex supplied an inductive proof that the $(n + 1)$ st join power of Fin_2 is equivalent to S^n by demonstrating a more general equivalence $\Sigma X \simeq X * \text{Fin}_2$. Thus, the desired equivalence could be understood as an instance of “join associativity”, an equivalence between $((\text{Fin}_2 * \text{Fin}_2) * \text{Fin}_2) * \text{Fin}_2$ and $(\text{Fin}_2 * \text{Fin}_2) * (\text{Fin}_2 * \text{Fin}_2)$.

Rather than tackle this directly, Codex insisted on pursuing various side quests until directed more clearly to stop. The prompter finally came up with the right combination

of words to get Codex to persist in its work, though this was to no avail. A first attempt at a direct comparison equivalence $A * (B * C) \simeq (A * B) * C$ stalled after several hours. The next day, Codex was asked to pivot to arguing by universal properties, though again this was unsuccessful. The following day, after this failed, the third author prompted Codex to look at the Coq-HoTT formalization, which compared both $A * (B * C)$ and $(A * B) * C$ to a natively-defined triple join. This took two days of work to implement but was completed successfully on June 25th.

At this point, the only remaining result was the Blakers–Massey theorem and its corollary, the Freudenthal suspension theorem. While Codex initially proposed a direct proof of the isomorphism $\pi_3 S^3 \cong \pi_2 S^2$ needed for our specific goal, the third author redirected it with a prompt:

I’ve changed my mind. Give me a plan instead that goes after a library quality Freudenthal suspension theorem for general use in agda-unimath together with the special case needed here. What is needed for this?

Over seven hours of repeated prompting, culminating in the instruction “Keep working hard and don’t stop until the Blakers Massey theorem is proven,” Codex successfully autoformalized a proof of the Blakers–Massey theorem, guided primarily by the Coq-HoTT library, in 5000+ lines of code.

This final step was started and completed on June 25th, and the unconditional isomorphism $\pi_3 S^2 \cong \mathbb{Z}$ was autoformalized 30 minutes later, after a handoff to a different machine. Afterwards Codex was asked to prove that $\pi_n S^n \cong \mathbb{Z}$ for all positive n , and work on polishing up the long exact sequence of the fibration, for instance extending the exactness of pointed sets on the path components $\pi_0 F \rightarrow \pi_0 E \rightarrow \pi_0 B$. This final step of expanding the development of the long exact sequence took place on June 26th and June 27th.

3.5 Metrics The formalization comprises 87 literate Agda files and 31,464 lines under the project’s source directory, of which about 28,000 are Agda code and the rest surrounding mathematical prose. Nearly all of it is contained in the synthetic homotopy theory namespace. The two largest files are the join associativity development (7,436 lines), which was the hardest-fought result of §3.4, and the Blakers–Massey theorem (5,474 lines), which was completed surprisingly quickly, most likely due to the learned route of consulting the Coq-HoTT library for guidance and marginal improvements in user prompting. The repository records 151 commits between June 3rd and June 27th, when the autoformalized Agda code was written. Later commits were made in preparing the repository for public release, generating the Agda MCP user experience report, fixing bugs in the check.sh file, cleaning up the documentation concerning the long exact sequence, and adding anonymized versions of the raw codex logs.

The total amount of tokens used during June 9th – June 27th by the third author was 2,079,055,195, for a cost of \$1,750.49. As noted above, the prompting done from June 3rd – June 8th was inadvertently done using a free chatGPT account, from which precise usage records cannot be recovered. This period included the first 11 commits

on June 3rd, three commits on June 4th, and seven commits on June 7th (including two by the fourth author, who similarly failed to set up the intended API key).

4 CONCLUSIONS

We conclude by offering some lessons from our experiment and also describing plans for further work.

4.1 Takeaways We credit the success of this autoformalization experiment to various sources.

Firstly, the `agda-unimath` library contained a lot of useful background homotopy theory and group theory, including the calculation that $\pi_1 S^1 \cong \mathbb{Z}$ together with various other infrastructure related to the circle, as well as general results about loop spaces, suspensions, joins, pushouts, descent, truncation levels, and infrastructure relating abstract and concrete groups. In addition, it had clearly written guides for contributors and an easily searchable library with a uniform style.

Secondly, there was a clearly-written natural language source provided by [Uni13] for the material missing from the `agda-unimath` library, written using then novel terminology that has since become standard for the field. This appeared to support natural language reasoning to provide outlines for the general homotopy type theoretic proofs. These first two ingredients were the ones that led the authors to believe that this result was within scope of Codex.

We did not anticipate how important the existing formalizations in the Coq-HoTT library were to overcoming the bottlenecks described above. Given how central these human-authored guides appeared to be for the long exact sequence, the join associativity, and the proof of Blakers–Massey, we now believe that success of this experiment was ultimately predicated on the fact that humans have done this before. Thus, we feel that it is important to credit the human authors of the formalizations that guided Codex.

Mike Shulman formalized the long exact sequence in Coq-HoTT via what Codex considered the “hard structural route” of packaging the boundary map $\Omega B \rightarrow_* F$ of a fiber sequence $F \rightarrow_* E \rightarrow_* B$ as a map in its own iterated fiber sequences.

Dan Christensen formalized the join associativity in Coq-HoTT using several tricks that Codex copied, including formalizing the final equivalence as a composite involving two instances of join symmetry surrounding a “twisting” equivalence:

$$(A * B) * C \simeq C * (A * B) \simeq A * (C * B) \simeq A * (B * C).$$

Mike Shulman formalized Blakers–Massey in Coq-HoTT following the proof of [HFLL16; ABFJ20], but using a novel construction that he describes in the pull request¹⁰. Again, Codex copied this construction, specializing its work from an arbitrary modality to the truncation modalities.

Our initial impression is that Codex is capable of translating formalized source code from one library to another. The incompatibility of formalized mathematics across different libraries has been an important practical problem within the community of

¹⁰github.com/HoTT/Coq-HoTT/pull/1078

formalizers. The capacity of Codex to transfer code from one library to another seems to be a major step forward in this regard.

4.2 Future technical improvements The report we prepared for the maintainers of the Agda MCP server, included in the project repository together with its extraction script and the redacted evidence for every suspicious call, traces the most serious failures to a single design problem: the server’s replies conflate three separate questions — whether the request was processed, whether Agda accepted the command, and whether any goals, metavariables, or constraints remain open. An agent that takes such a reply at face value can assert a proof that Agda rejects. As our attempts to contact the maintainer of that repository have failed, we plan to develop an improved server, tested against the defects catalogued in the report, with this project’s formalization as its first test corpus. If interactive feedback can be made trustworthy in this way, a full run of Agda, as in `check.sh`, would be needed only as a final safeguard.

Such a server would matter beyond convenience because of the way Agda is used. An Agda proof is a program, ordinarily written incrementally through typed holes: a partially written proof can be checked with its gaps left open, and at each gap Agda reports the goal and the available hypotheses. A proof advances by inspecting a hole, splitting it into cases, or filling it, and the MCP server exposes to an agent exactly the interaction points that a human sees in the editor. A faithful server would therefore give a human and an agent a shared, machine-checked view of the same unfinished proof, in which either one can leave a hole for the other to attempt — the top-down style of collaboration suggested to us by Andrej Bauer. The division of labor would then be visible in the proof state itself and checked by Agda at every step, rather than coordinated through instructions to the agent. We consider this one of the more promising directions to emerge from the experiment.

A second improvement would automate the prompting itself: a driver that hands the agent one module of the formalization plan at a time and accepts each result only after independent verification, so that progress would not depend on the availability of a human prompter (§3.4). We leave this to future work.

A further place we would like to improve is by giving the model better coaching regarding its communications with us. For instance, Claude Code also generated a section on “Tractability and risks”, describing potential pitfalls of the project. It was able to identify sections of the formalization that would be high-effort due to a large amount of definitions or framework missing from `agda-unimath`. Curiously, it did so using a mode of speaking not traditionally found among mathematical prose. To highlight one such example:

“Hopf fibration risk: Constructing the fibration from the H-space circle and proving the total space is S^3 will stress pushout, join, and fiberwise-descent infrastructure.”

We also found that Codex defaulted to use a similar mode of speaking in the logs we asked it to maintain. Since we are not accustomed to such writing, it was much

harder for us to understand, and we did not find the various AI-generated logs to be as useful as we would have liked.

4.3 Library impacts As noted in the introduction, we do not consider an autoformalization project to be completed until it makes a commensurable contribution to the supporting library, in this case agda-unimath. The raw Codex generated code in our repository does not yet meet the standards for an agda-unimath PR, much less the standards of the library. It is insufficiently documented, insufficiently explained, and far too verbose. The formalization code also needs thorough review by us first, and later by library maintainers.

One motivation for writing this report is to facilitate communication with the maintainers of the agda-unimath library. We hope to discuss the possibility of extracting library-quality code to contribute to the library over a series of PRs in a way that is sensitive to the amount of human labor that the review process will require, in particular by contributing our own efforts to the refereeing process. As we view this to be of paramount importance to the evolving interactions between generative AI and computer formalization of mathematics, we plan to update this preprint with the results of our planned future conversations with the broader agda-unimath community.

REFERENCES

- [ABFJ20] Mathieu Anel, Georg Biedermann, Eric Finster, and André Joyal. “A generalized Blakers–Massey theorem”. In: *Journal of Topology* 13.4 (2020), pp. 1521–1553. DOI: [10.1112/topo.12163](https://doi.org/10.1112/topo.12163). <https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/topo.12163>.
- [Avi26] Jeremy Avigad. *Mathematicians in the age of AI*. 2026. arXiv: [2603.03684](https://arxiv.org/abs/2603.03684) [[math.HO](https://arxiv.org/abs/2603.03684)]. <https://arxiv.org/abs/2603.03684>.
- [AKL15] Jeremy Avigad, Krzysztof Kapulkin, and Peter LeFanu Lumsdaine. “Homotopy Limits in Type Theory”. In: *Mathematical Structures in Computer Science* 25.5 (2015), pp. 1040–1070. DOI: [10.1017/S0960129514000498](https://doi.org/10.1017/S0960129514000498). arXiv: [1304.0680](https://arxiv.org/abs/1304.0680).
- [AW09] Steve Awodey and Michael A. Warren. “Homotopy Theoretic Models of Identity Types”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 146.1 (2009), pp. 45–55. DOI: [10.1017/S0305004108001783](https://doi.org/10.1017/S0305004108001783). arXiv: [0709.0248](https://arxiv.org/abs/0709.0248).
- [BGLS+17] Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Michael Shulman, et al. “The HoTT library: a formalization of homotopy type theory in Coq”. In: *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*. CPP 2017. Paris, France: Association for Computing Machinery, 2017, pp. 164–172. ISBN: 9781450347051. DOI: [10.1145/3018610.3018615](https://doi.org/10.1145/3018610.3018615).
- [Ben79] L. S. van Benthem Jutting. *Checking Landau’s Grundlagen in the AUTOMATH system*. Vol. 83. Mathematisch Centre Tracts. Mathematisch Centrum, Amsterdam, 1979, pp. iv+120. ISBN: 90-6196-147-5.

- [BBCDG26] Marc Bezem, Ulrik Buchholtz, Pierre Cagne, Bjørn Ian Dundas, and Daniel R. Grayson. *Symmetry*. Commit 10bb0f6. Aug. 13, 2026. <https://github.com/UniMath/SymmetryBook/tree/10bb0f6>.
- [BCFR25] Ulrik Buchholtz, J. Daniel Christensen, Jarl G. Taxerås Flaten, and Egbert Rijke. “Central H-Spaces and Banded Types”. In: *Journal of Pure and Applied Algebra* 229.6, 107963 (2025). DOI: [10.1016/j.jpaa.2025.107963](https://doi.org/10.1016/j.jpaa.2025.107963). arXiv: [2301.02636](https://arxiv.org/abs/2301.02636).
- [BDR18] Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. “Higher Groups in Homotopy Type Theory”. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 205–214. DOI: [10.1145/3209108.3209150](https://doi.org/10.1145/3209108.3209150).
- [BR18] Ulrik Buchholtz and Egbert Rijke. “The Cayley–Dickson Construction in Homotopy Type Theory”. In: *Higher Structures* 2.1 (2018), pp. 30–41. arXiv: [1610.01134](https://arxiv.org/abs/1610.01134).
- [Mat20] The mathlib Community. “The Lean Mathematical Library”. In: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2020. New Orleans, LA, USA: ACM, Jan. 2020, pp. 367–381. DOI: [10.1145/3372885.3373824](https://doi.org/10.1145/3372885.3373824).
- [GAAB+13] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, et al. “A Machine-Checked Proof of the Odd Order Theorem”. In: *LNCS*. Ed. by Sandrine Blazy, Christine Paulin, and David Pichardie. Vol. 7998. LNCS. Rennes, France: Springer, July 2013, pp. 163–179. DOI: [10.1007/978-3-642-39634-2_14](https://doi.org/10.1007/978-3-642-39634-2_14). <https://inria.hal.science/hal-00816699>.
- [HABD+17] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, et al. “A Formal Proof of the Kepler Conjecture”. In: *Forum of Mathematics, Pi* 5, e2 (2017). DOI: [10.1017/fmp.2017.1](https://doi.org/10.1017/fmp.2017.1).
- [HFLL16] Kuen-Bang Hou (Favonia), Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine. “A Mechanization of the Blakers–Massey Connectivity Theorem in Homotopy Type Theory”. In: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. Ed. by Martin Grohe, Eric Koskinen, and Natarajan Shankar. LICS ’16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 565–574. ISBN: 978-1-4503-4391-6. DOI: [10.1145/2933575.2934545](https://doi.org/10.1145/2933575.2934545).
- [Joh] Niles Johnson. *A visualization of the Hopf fibration*. <https://nilesjohnson.net/hopf.html> (visited on 08/14/2026).
- [Kor08] Artur Korniłowicz. “A Proof of the Jordan Curve Theorem via the Brouwer Fixed Point Theorem”. In: 2008. <https://api.semanticscholar.org/CorpusID:15229462>.
- [Lea] Lean Community. *Lean AI Formalization Leaderboard*. <https://lean-lang.org/eval/> (visited on 08/14/2026).
- [Lic13] Daniel R. Licata. *Homotopy Theory in Type Theory: Progress Report*. May 20, 2013. <https://homotopytypetheory.org/2013/05/20/homotopy-theory-in-type-theory-progress-report/> (visited on 08/14/2026).

- [LS13] Daniel R. Licata and Michael Shulman. “Calculating the Fundamental Group of the Circle in Homotopy Type Theory”. In: *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2013, pp. 223–232. DOI: [10.1109/LICS.2013.28](https://doi.org/10.1109/LICS.2013.28). arXiv: [1301.3443](https://arxiv.org/abs/1301.3443).
- [Lur03] Jacob Lurie. *On Infinity Topoi*. 2003. arXiv: [math/0306109](https://arxiv.org/abs/math/0306109) [math.CT]. <https://arxiv.org/abs/math/0306109>.
- [Mor] Kim Morrison. π_3 of the 2-sphere is $\mathbb{Z}$. Lean Focused Research Organization. https://lean-lang.org/eval/problems/pi3_sphere_two_mulEquiv_int/ (visited on 08/14/2026).
- [Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Princeton, NJ: Institute for Advanced Study, 2013. <https://homotopytypetheory.org/book>.
- [Rij25] Egbert Rijke. *Introduction to Homotopy Type Theory*. Cambridge Studies in Advanced Mathematics 219. Cambridge University Press, 2025. ISBN: 9781108844161. DOI: [10.1017/9781108933568](https://doi.org/10.1017/9781108933568).
- [RSPB+21] Egbert Rijke, Elisabeth Stenholm, Jonathan Prieto-Cubides, Fredrik Bakke, et al. *The agda-unimath library*. 2021. <https://github.com/UniMath/agda-unimath/> (visited on 08/14/2026).
- [Shu15] Michael Shulman. *The Univalent Perspective on Classifying Spaces*. Jan. 19, 2015. https://golem.ph.utexas.edu/category/2015/01/the_univalent_perspective_on_c.html (visited on 08/14/2026).
- [Shu19] Michael Shulman. *All $(\infty, 1)$ -toposes have strict univalent universes*. 2019. arXiv: [1904.07004](https://arxiv.org/abs/1904.07004) [math.AT]. <https://arxiv.org/abs/1904.07004>.
- [Shu24] Michael Shulman. “Strange New Universes: Proof Assistants and Synthetic Foundations”. In: *Bulletin of the American Mathematical Society*. New Series 61.2 (Apr. 2024), pp. 257–270. DOI: [10.1090/bull/1830](https://doi.org/10.1090/bull/1830).
- [Voe10] Vladimir Voevodsky. *Univalent Basics. Part of the Foundations Library for Coq*. UniMath, Feb. 2010–Sept. 2011. <https://github.com/UniMath/Foundations/blob/master/Generalities/uu0.v> (visited on 08/14/2026).
- [Voe14] Vladimir Voevodsky. “On the origins and motivations of univalent foundations”. In: *The Institute Letter Summer 2014* (2014). <https://www.ias.edu/ideas/2014/voevodsky-origins>.

JOHNS HOPKINS UNIVERSITY, 3400 N CHARLES STREET, BALTIMORE, MD 21218

Email address: dcarran3@jhu.edu

Email address: cliu238@jhu.edu

Email address: eriehl@jhu.edu

Email address: erijke1@jhu.edu