Carnegie Mellon University

Verification, Formalisation and Canonisation: Formal Theorem Proving in the Age of AI

Tata Institute of Fundamental Research

Sidharth Hariharan

6 August 2026

sidharthhariharan@cmu.edu

Contents

Formalisation Before AI

A Brief History of Interactive Theorem Provers

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project
- **1984:** T Coquand and Huet implement the Calculus of Constructions at INRIA-Rocquencourt.

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project
- **1984:** T Coquand and Huet implement the Calculus of Constructions at INRIA-Rocquencourt.
- **1986:** Paulson launches **Isabelle**, based on simple type theory

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project
- **1984:** T Coquand and Huet implement the Calculus of Constructions at INRIA-Rocquencourt.
- **1986:** Paulson launches **Isabelle**, based on simple type theory
- **1989:** T Coquand and Paulin expands Coquand's implementation to the Calculus of Inductive Constructions, leading to the first version of **Coq** (recently renamed **Rocq**).

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project
- **1984:** T Coquand and Huet implement the Calculus of Constructions at INRIA-Rocquencourt.
- **1986:** Paulson launches **Isabelle**, based on simple type theory
- **1989:** T Coquand and Paulin expands Coquand's implementation to the Calculus of Inductive Constructions, leading to the first version of **Coq** (recently renamed **Rocq**).
- **1996-98:** C Coquand implements the original version of Agda in Haskell

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project
- **1984:** T Coquand and Huet implement the Calculus of Constructions at INRIA-Rocquencourt.
- **1986:** Paulson launches **Isabelle**, based on simple type theory
- **1989:** T Coquand and Paulin expands Coquand's implementation to the Calculus of Inductive Constructions, leading to the first version of **Coq** (recently renamed **Rocq**).
- **1996-98:** C Coquand implements the original version of Agda in Haskell
- **2004:** Earliest formalisation I can find in the **Archive of Formal Proofs**

A Brief History of Interactive Theorem Provers

- **1967:** de Bruijn creates **Automath**, which implements propositions as types
- **1973:** Trybulec gives an influential presentation leading to the development of **Mizar**, which has a set-theoretic foundation
- **1984:** Constable et al launch **NuPRL**, which led to the PRL ((**P**roof/**P**rogram Refinement Logic) project) project
- **1984:** T Coquand and Huet implement the Calculus of Constructions at INRIA-Rocquencourt.
- **1986:** Paulson launches **Isabelle**, based on simple type theory
- **1989:** T Coquand and Paulin expands Coquand's implementation to the Calculus of Inductive Constructions, leading to the first version of **Coq** (recently renamed **Rocq**).
- **1996-98:** C Coquand implements the original version of Agda in Haskell
- **2004:** Earliest formalisation I can find in the **Archive of Formal Proofs**
- **2013:** de Moura creates the first version of **Lean**

21st Century Formalisation Milestones

21st Century Formalisation Milestones

2003 — Hales launches **Flyspeck**: a formal proof of the Kepler conjecture

21st Century Formalisation Milestones

2003 Hales launches **Flyspeck**: a formal proof of the Kepler conjecture

2004 **Prime Number Theorem** (elementary proof) formalised in Isabelle (Avigad, Donnelly, Gray, Raff [1]), and the analytic proof in HOL Light in 2009 (Harrison [12])

2005 **Four Colour Theorem** in Coq: Gonthier, with Werner [6]. **Jordan Curve Theorem** in HOL Light: Hales [11]

21st Century Formalisation Milestones

2003 Hales launches **Flyspeck**: a formal proof of the Kepler conjecture

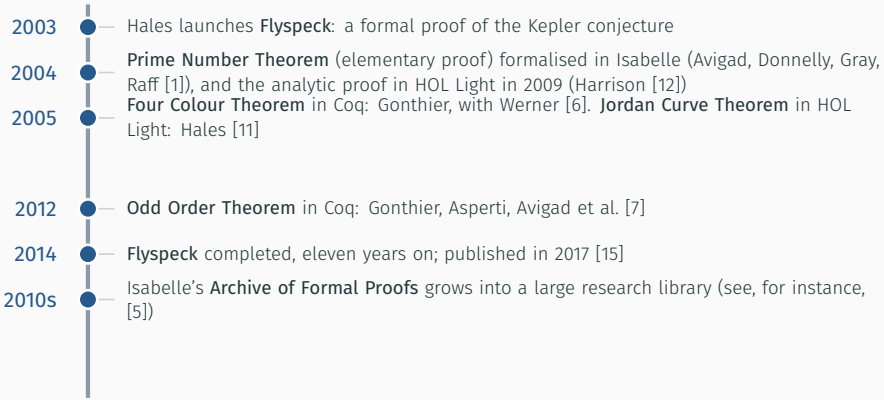
2004 **Prime Number Theorem** (elementary proof) formalised in Isabelle (Avigad, Donnelly, Gray, Raff [1]), and the analytic proof in HOL Light in 2009 (Harrison [12])

2005 **Four Colour Theorem** in Coq: Gonthier, with Werner [6]. **Jordan Curve Theorem** in HOL Light: Hales [11]

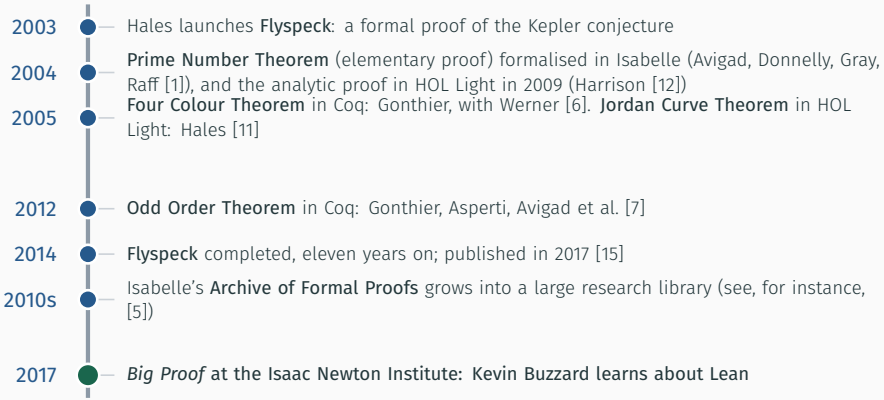
2012 **Odd Order Theorem** in Coq: Gonthier, Asperti, Avigad et al. [7]

2014 **Flyspeck** completed, eleven years on; published in 2017 [15]

21st Century Formalisation Milestones



21st Century Formalisation Milestones



Lean Takes Off (aka the Kevin Buzzard et al era)

Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]

Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]
- **2020** — *Sphere eversion*: Massot, Nash, van Doorn — Smale's theorem

Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]
- **2020** — *Sphere eversion*: Massot, Nash, van Doorn — Smale's theorem
- **2020–22** — *Liquid Tensor Experiment*: Scholze's challenge, answered by Commelin, Topaz et al. [4]

Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]
- **2020** — *Sphere eversion*: Massot, Nash, van Doorn — Smale's theorem
- **2020–22** — *Liquid Tensor Experiment*: Scholze's challenge, answered by Commelin, Topaz et al. [4]
- **2023** — *Exponential improvement for diagonal Ramsey*: Mehta formalises a fifty-page preprint of Campos, Griffiths, Morris and Sahasrabudhe in five months [14]

Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]
- **2020** — *Sphere eversion*: Massot, Nash, van Doorn — Smale's theorem
- **2020–22** — *Liquid Tensor Experiment*: Scholze's challenge, answered by Commelin, Topaz et al. [4]
- **2023** — *Exponential improvement for diagonal Ramsey*: Mehta formalises a fifty-page preprint of Campos, Griffiths, Morris and Sahasrabudhe in five months [14]
- **2024** — *PNT+* launched by Kontorovich and Tao

Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]
- **2020** — *Sphere eversion*: Massot, Nash, van Doorn — Smale's theorem
- **2020–22** — *Liquid Tensor Experiment*: Scholze's challenge, answered by Commelin, Topaz et al. [4]
- **2023** — *Exponential improvement for diagonal Ramsey*: Mehta formalises a fifty-page preprint of Campos, Griffiths, Morris and Sahasrabudhe in five months [14]
- **2024** — *PNT+* launched by Kontorovich and Tao
- **2024** — *Sphere Packing* launched by Hariharan and Viazovska
- **2024** — *Fermat's Last Theorem* launched by Buzzard

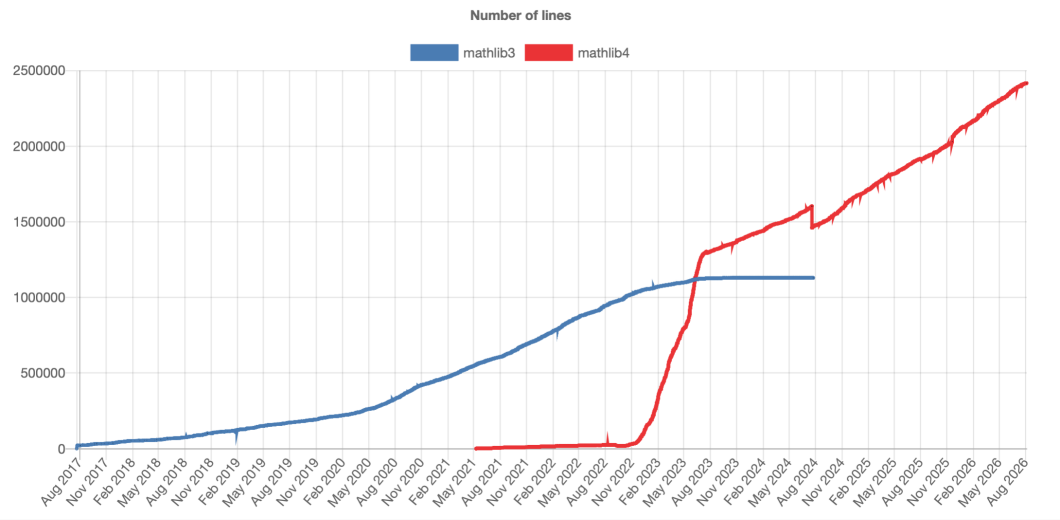
Lean Takes Off (aka the Kevin Buzzard et al era)

- **2019** — *Perfectoid spaces*: Buzzard, Commelin, Massot [2]
- **2020** — *Sphere eversion*: Massot, Nash, van Doorn — Smale's theorem
- **2020–22** — *Liquid Tensor Experiment*: Scholze's challenge, answered by Commelin, Topaz et al. [4]
- **2023** — *Exponential improvement for diagonal Ramsey*: Mehta formalises a fifty-page preprint of Campos, Griffiths, Morris and Sahasrabudhe in five months [14]
- **2024** — *PNT+* launched by Kontorovich and Tao
- **2024** — *Sphere Packing* launched by Hariharan and Viazovska
- **2024** — *Fermat's Last Theorem* launched by Buzzard

… and many, many more…

The Development of Mathlib

The Development of Mathlib



Taken from `https://leanprover-community.github.io/mathlib_stats.html`

Key Features of Mathlib

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

- Theorems are stated in maximal (resonable) generality.

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

- Theorems are stated in maximal (resonable) generality.
- Definitions are designed to be flexible enough to accommodate generality while still serving the desired purpose.

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

- Theorems are stated in maximal (resonable) generality.
- Definitions are designed to be flexible enough to accommodate generality while still serving the desired purpose.
- Each definition is accompanied by extensive API.

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

- Theorems are stated in maximal (resonable) generality.
- Definitions are designed to be flexible enough to accommodate generality while still serving the desired purpose.
- Each definition is accompanied by extensive API.
- Commonly used techniques in proofs are combined into tactics.

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

- Theorems are stated in maximal (resonable) generality.
- Definitions are designed to be flexible enough to accommodate generality while still serving the desired purpose.
- Each definition is accompanied by extensive API.
- Commonly used techniques in proofs are combined into tactics.
- Frameworks and theories are strongly encouraged where possible.

Key Features of Mathlib

Mathlib seeks to be highly optimised for downstream use. To that end:

- Theorems are stated in maximal (resonable) generality.
- Definitions are designed to be flexible enough to accommodate generality while still serving the desired purpose.
- Each definition is accompanied by extensive API.
- Commonly used techniques in proofs are combined into tactics.
- Frameworks and theories are strongly encouraged where possible.
- Naming and style conventions are strictly enforced.

AI and the Formal Turn

Why AI cares about Formalisation

Why AI cares about Formalisation

- Verified training data

Why AI cares about Formalisation

- Verified training data
- Instant feedback from a compiler

Autoformalisation

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy
3. The end product of the Z code written by Model X corresponds semantically to the contents of Y

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy
3. The end product of the Z code written by Model X corresponds semantically to the contents of Y

An autoformalisation could either involve translating a mathematical endeavour from natural language to Z or performing the entire endeavour in Z.

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy
3. The end product of the Z code written by Model X corresponds semantically to the contents of Y

An autoformalisation could either involve translating a mathematical endeavour from natural language to Z or performing the entire endeavour in Z. Often both.

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy
3. The end product of the Z code written by Model X corresponds semantically to the contents of Y

An autoformalisation could either involve translating a mathematical endeavour from natural language to Z or performing the entire endeavour in Z. Often both.

The benefit of the first mode is that it removes the requirement of learning Z to avail its benefits.

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy
3. The end product of the Z code written by Model X corresponds semantically to the contents of Y

An autoformalisation could either involve translating a mathematical endeavour from natural language to Z or performing the entire endeavour in Z. Often both.

The benefit of the first mode is that it removes the requirement of learning Z to avail its benefits.

The benefit of the second mode is that it gives us confidence in the machine output.

Autoformalisation

Autoformalisation is when a machine writes up a proof in a formal theorem prover.

When we say “Model X has autoformalised (Theorem/Proof/Definition) Y in Theorem Prover Z”, we typically mean

1. Model X wrote Z code that compiles
2. Model X did so with a reasonable degree of autonomy
3. The end product of the Z code written by Model X corresponds semantically to the contents of Y

An autoformalisation could either involve translating a mathematical endeavour from natural language to Z or performing the entire endeavour in Z. Often both.

The benefit of the first mode is that it removes the requirement of learning Z to avail its benefits.

The benefit of the second mode is that it gives us confidence in the machine output.

MODULO SEMANTIC CORRESPONDENCE (and correctness of Z)

Models that can Formalise

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO
- **Aristotle:** User-facing Lean 4 autoformalisation model

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO
- **Aristotle:** User-facing Lean 4 autoformalisation model
- **Gauss:** Demonstrates that models can write very large volumes of compiling Lean code

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO
- **Aristotle:** User-facing Lean 4 autoformalisation model
- **Gauss:** Demonstrates that models can write very large volumes of compiling Lean code
- **AxiomProver:** Used to solve a number of open problems

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO
- **Aristotle:** User-facing Lean 4 autoformalisation model
- **Gauss:** Demonstrates that models can write very large volumes of compiling Lean code
- **AxiomProver:** Used to solve a number of open problems
- AlephProver, Kimina Prover, Seed-Prover, AxProver, etc etc etc

Models that can Formalise

A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO
- **Aristotle:** User-facing Lean 4 autoformalisation model
- **Gauss:** Demonstrates that models can write very large volumes of compiling Lean code
- **AxiomProver:** Used to solve a number of open problems
- AlephProver, Kimina Prover, Seed-Prover, AxProver, etc etc etc

Off-the-shelf LLMs have also become quite good at formalisation.

Models that can Formalise

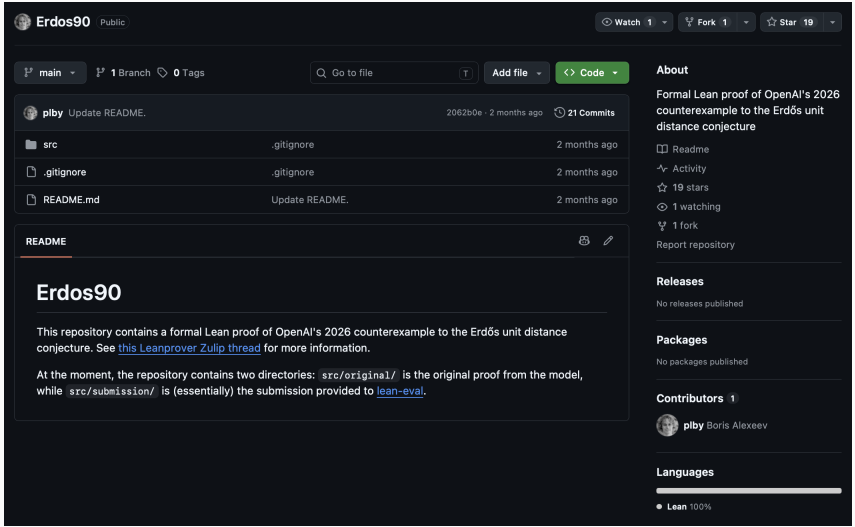
A number of models have been created solely for the purpose of autoformalisation.

- **Évariste:** Early model developed at Meta AI
- **AlphaProof:** Achieves silver-level performance at the IMO
- **Aristotle:** User-facing Lean 4 autoformalisation model
- **Gauss:** Demonstrates that models can write very large volumes of compiling Lean code
- **AxiomProver:** Used to solve a number of open problems
- AlephProver, Kimina Prover, Seed-Prover, AxProver, etc etc etc

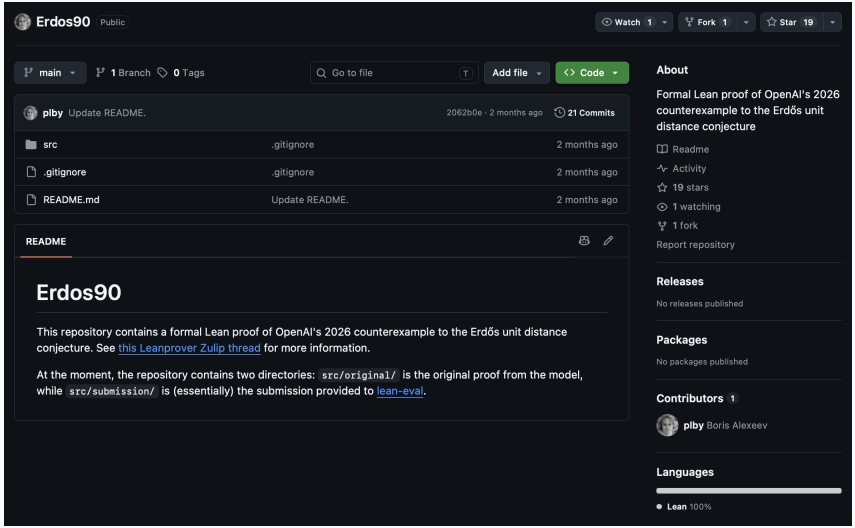
Off-the-shelf LLMs have also become quite good at formalisation. Skills make them even better.

The Unit Distance Conjecture

The Unit Distance Conjecture

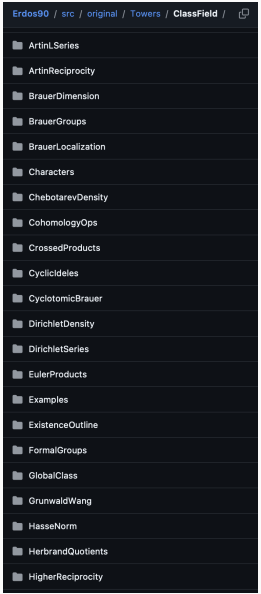


The Unit Distance Conjecture

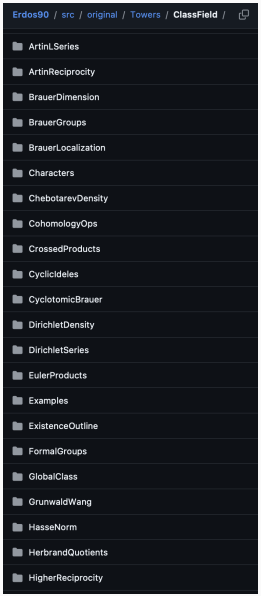


The Unit Distance Conjecture

The Unit Distance Conjecture



The Unit Distance Conjecture




```
abbrev ElementaryAbelianGroup (n : ℕ) : Type :=
  Fin n → Multiplicative (ZMod 3)

/-- A field is a cyclic cubic extension of `Q` if it is Galois, cubic,
and its Galois group is cyclic. -/
def CyclicCubicQ
    (K : Type) [Field K] [NumberField K] [Algebra ℚ K] : Prop :=
  Module.finrank ℚ K = 3 ∧ IsGalois ℚ K ∧ IsCyclic (Gal(K/ℚ))

/-- A field is ramified only at the primes in `S` if every prime outside `S`
is unramified in its ring of integers. -/
def RamifiedOnlyAt
    (K : Type) [Field K] [NumberField K] [Algebra ℚ K] (S : Finset ℕ) : Prop :=
  ∀ q, Nat.Prime q → q ∉ S →
    RationalPrimeUnramified (S := NumberField.RingOfIntegers K) q

/-- A `Q`-field embedding between two number fields. -/
def EmbedsIntoField
    (K L : Type) [Field K] [NumberField K] [Algebra ℚ K]
    [Field L] [NumberField L] [Algebra ℚ L] : Prop :=
  Nonempty (K →ₐ[ℚ] L)

/-- A `Q`-field embedding of a finite number field into a blueprint extension. -/
def EmbedsIntoExtension
    (K : Type) [Field K] [NumberField K] [Algebra ℚ K]
    (L : Type) [Field L] [Algebra ℚ L] : Prop :=
  Nonempty (K →ₐ[ℚ] L)

/-- A `Q`-field embedding between two blueprint extensions. -/
def ExtensionEmbeds
    (K L : Type) [Field K] [Algebra ℚ K] [Field L] [Algebra ℚ L] : Prop :=
  Nonempty (K →ₐ[ℚ] L)

/-- `E` is a compositum of the family `L` if every `L i` embeds into `E` and `E` is initial
among such overfields. This is still an abstract universal property, but it now talks directly
about raw field types rather than hiding them inside a wrapper. -/
def CompositumFamily
    {ι : Type} (L : ι → Type) [∀ i, Field (L i)] [∀ i, NumberField (L i)]
    [∀ i, Algebra ℚ (L i)]
    (E : Type) [Field E] [NumberField E] [Algebra ℚ E] : Prop :=
  (∀ i, EmbedsIntoField (L i) E) ∧
    ∀ (F : Type) [Field F] [NumberField F] [Algebra ℚ F],
      (∀ i, EmbedsIntoField (L i) F) → EmbedsIntoField E F
```

A Distinction With a Difference

A History of Building Formal Infrastructure

A History of Building Formal Infrastructure

In the pre-AI era, formalisation was synonymous with building formal infrastructure.

A History of Building Formal Infrastructure

In the pre-AI era, formalisation was synonymous with building formal infrastructure.

For instance, the HOL Light Theory of Euclidean Spaces was, “as planned[,] applied to the Flyspeck project and has become the basis of a significant development of results in complex analysis, among others.” [13]

A History of Building Formal Infrastructure

In the pre-AI era, formalisation was synonymous with building formal infrastructure.

For instance, the HOL Light Theory of Euclidean Spaces was, “as planned[,] applied to the Flyspeck project and has become the basis of a significant development of results in complex analysis, among others.” [13]

Similarly, Gonthier et al “developed a comprehensive set of reusable libraries of formalized mathematics, including results in finite group theory, linear algebra, Galois theory, and the theories of the real and complex algebraic numbers” to support their formalisation of Feit-Thompson [7].

A History of Building Formal Infrastructure

In the pre-AI era, formalisation was synonymous with building formal infrastructure.

For instance, the HOL Light Theory of Euclidean Spaces was, “as planned[,] applied to the Flyspeck project and has become the basis of a significant development of results in complex analysis, among others.” [13]

Similarly, Gonthier et al “developed a comprehensive set of reusable libraries of formalized mathematics, including results in finite group theory, linear algebra, Galois theory, and the theories of the real and complex algebraic numbers” to support their formalisation of Feit-Thompson [7].

Of Avigad et al’s Isabelle formalisation of the PNT (following Selberg) [1], Harrison writes, “Avigad et al. were motivated to develop general libraries that could be of use in other contexts. In particular, their paper presents a general formalization of ‘big O’ asymptotic expressions.” [12]

Revisiting Motivations

Revisiting Motivations

In the Flyspeck announcement, Hales wrote, “my motivations for this project are far more complex than a simple hope of removing residual doubt from the minds of [a] few referees. Indeed, I see formal methods as fundamental to the long-term growth of mathematics.” [10]

Revisiting Motivations

In the Flyspeck announcement, Hales wrote, “my motivations for this project are far more complex than a simple hope of removing residual doubt from the minds of [a] few referees. Indeed, I see formal methods as fundamental to the long-term growth of mathematics.” [10]

Indeed, in a 2024 retrospective, Hales wrote, “[John Harrison] contributed libraries in trigonometry, multivariate differential calculus, measure and integration in $\mathbb{R}^n$, convex sets, and polyhedra. He contributed powerful tactics such as the without-loss-of-generality tactic, which automates some of what mathematicians mean in proofs when they write those words.” [9]

Revisiting Motivations

In the Flyspeck announcement, Hales wrote, “my motivations for this project are far more complex than a simple hope of removing residual doubt from the minds of [a] few referees. Indeed, I see formal methods as fundamental to the long-term growth of mathematics.” [10]

Indeed, in a 2024 retrospective, Hales wrote, “[John Harrison] contributed libraries in trigonometry, multivariate differential calculus, measure and integration in $\mathbb{R}^n$, convex sets, and polyhedra. He contributed powerful tactics such as the without-loss-of-generality tactic, which automates some of what mathematicians mean in proofs when they write those words.” [9]

But Flyspeck, like any project, had a definitive goal. In the next paragraph, Hales wrote, “Aside from Harrison’s libraries, most lemmas were formulated minimally rather than in generality, because our main interest was the narrow goal of formalizing the Kepler conjecture rather than general library development. For example, the formalization avoids matrices, integration on manifolds (such as a sphere), and the general Jordan curve theorem (using only a special case).”

The Two Families of Formalisation Projects

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing).

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The second family is projects that were motivated by **correctness**.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The second family is projects that were motivated by **correctness**. Examples include Flyspeck (HOL Light), Four Colour (Rocq), Liquid Tensor (Lean), Unit Distance (Lean).

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The second family is projects that were motivated by **correctness**. Examples include Flyspeck (HOL Light), Four Colour (Rocq), Liquid Tensor (Lean), Unit Distance (Lean).

Projects from both families are supported by extensive libraries of formal mathematics, and have, in turn, fed back into them.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The second family is projects that were motivated by **correctness**. Examples include Flyspeck (HOL Light), Four Colour (Rocq), Liquid Tensor (Lean), Unit Distance (Lean).

Projects from both families are supported by extensive libraries of formal mathematics, and have, in turn, fed back into them.

But a more conscious effort is needed for projects from the second family to do this than the first.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The second family is projects that were motivated by **correctness**. Examples include Flyspeck (HOL Light), Four Colour (Rocq), Liquid Tensor (Lean), Unit Distance (Lean).

Projects from both families are supported by extensive libraries of formal mathematics, and have, in turn, fed back into them.

But a more conscious effort is needed for projects from the second family to do this than the first.

I think that AI is currently pushing the growth of the second family much faster than the first.

The Two Families of Formalisation Projects

All our examples thus far fall into two distinct families.

The first family is projects that were motivated by **building infrastructure** and **assessing the reach of existing infrastructure**. Examples include PNT (Isabelle - elementary and HOL - analytic), PNT+ (Lean - ongoing), Feit-Thompson (Rocq), Perfectoid Spaces (Lean), Sphere Packing (Lean - ongoing). This family also contains smaller efforts that often don't get the full status of "project", such as explicit library development efforts like formalising Galois theory in Lean.

The second family is projects that were motivated by **correctness**. Examples include Flyspeck (HOL Light), Four Colour (Rocq), Liquid Tensor (Lean), Unit Distance (Lean).

Projects from both families are supported by extensive libraries of formal mathematics, and have, in turn, fed back into them.

But a more conscious effort is needed for projects from the second family to do this than the first.

I think that AI is currently pushing the growth of the second family much faster than the first.

I also think some in the AI community don't realise the first family is different from the second.

The Two Modes of Formalisation

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content.

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a fully compiling artefact expressed in Z

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a fully compiling artefact expressed in Z
- whose main result corresponds semantically to Y.

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a fully compiling artefact expressed in Z
- whose main result corresponds semantically to Y.

A "**Canonisation** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a fully compiling artefact expressed in Z
- whose main result corresponds semantically to Y.

A "**Canonisation** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a formal verification of Y in Z

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a fully compiling artefact expressed in Z
- whose main result corresponds semantically to Y.

A "**Canonisation** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a formal verification of Y in Z
- where both the main result and the intermediate artefacts are designed to be reusable, maintainable, general, flexible, extensible... (in other words, designed as per the principles of a library).

The Two Modes of Formalisation

Having made a distinction in motivation, we now make a distinction in content. The contents of this slide are motivated largely by talks given by Alex Kontorovich at the *AI for Maths and Open Science* event at the INI and the ICM and a talk by Terence Tao at the ICM.

A "**Formal Verification** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a fully compiling artefact expressed in Z
- whose main result corresponds semantically to Y.

A "**Canonisation** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to

- a formal verification of Y in Z
- where both the main result and the intermediate artefacts are designed to be reusable, maintainable, general, flexible, extensible... (in other words, designed as per the principles of a library).

A "**Formalisation** of (Theorem/Proof/Definition) Y in the Theorem Prover Z" will refer to either a formal verification or a canonisation.

Canonising a Verification: The Sphere Packing Project

The Sphere Packing Problem in Dimension 8

The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16].

The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

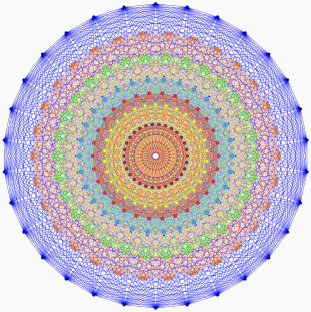
Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8].

The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8].

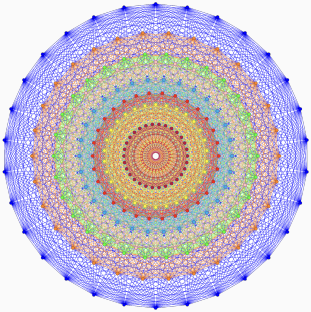


The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8]. Nodes are vectors and edges connect nearest neighbours.



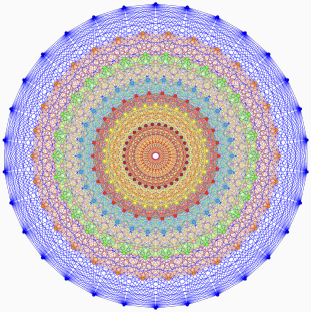
The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8]. Nodes are vectors and edges connect nearest neighbours.

This was long suspected, and in 2003, Henry Cohn and Noam Elkies proposed a method that many believed would offer a means to a proof.



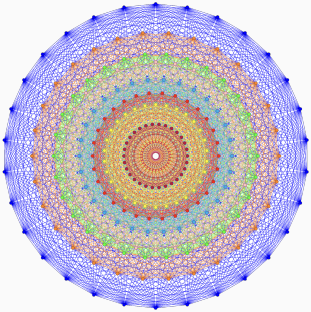
The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8]. Nodes are vectors and edges connect nearest neighbours.

This was long suspected, and in 2003, Henry Cohn and Noam Elkies proposed a method that many believed would offer a means to a proof. They showed that if a certain type of $\mathbb{R}^n \to \mathbb{R}$ function is constructed, then all sphere packings in $\mathbb{R}^n$ are bounded above by a certain quantity that depends on the function.



The Sphere Packing Problem in Dimension 8

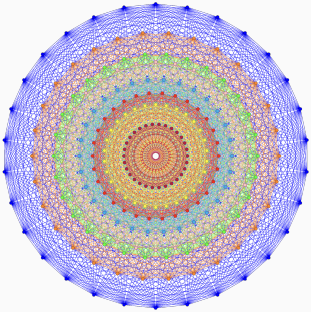
The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8]. Nodes are vectors and edges connect nearest neighbours.

This was long suspected, and in 2003, Henry Cohn and Noam Elkies proposed a method that many believed would offer a means to a proof. They showed that if a certain type of $\mathbb{R}^n \to \mathbb{R}$ function is constructed, then all sphere packings in $\mathbb{R}^n$ are bounded above by a certain quantity that depends on the function.

Unfortunately, constructing such functions is extremely difficult.



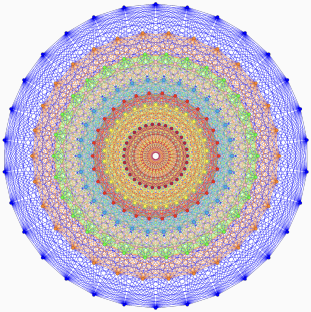
The Sphere Packing Problem in Dimension 8

The sphere packing problem is to answer the following question:

> What is the densest arrangement in $\mathbb{R}^n$ of non-overlapping n-spheres of the same radius?

Viazovska famously showed that the densest arrangement in $\mathbb{R}^8$ corresponds to spheres centred at points on the E_8 root lattice [16]. Below, we give an illustration of a projection of the E_8 root system onto its so-called Coxeter plane [8]. Nodes are vectors and edges connect nearest neighbours.

This was long suspected, and in 2003, Henry Cohn and Noam Elkies proposed a method that many believed would offer a means to a proof. They showed that if a certain type of $\mathbb{R}^n \to \mathbb{R}$ function is constructed, then all sphere packings in $\mathbb{R}^n$ are bounded above by a certain quantity that depends on the function.



Unfortunately, constructing such functions is extremely difficult. Viazovska's breakthrough was to construct such a function using the theory of modular forms.

Timeline

Timeline

28 February 2024 — First commit: SH and MV launch project

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Infrastructure on sphere packings, modular forms, and Fourier analysis

Blueprint refinements: SL's proof of the modular form inequalities

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Infrastructure on sphere packings, modular forms, and Fourier analysis

Blueprint refinements: SL's proof of the modular form inequalities

Dabbling in metaprogramming

Groundwork for constructing the magic function

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Infrastructure on sphere packings, modular forms, and Fourier analysis

Blueprint refinements: SL's proof of the modular form inequalities

Dabbling in metaprogramming

Groundwork for constructing the magic function

13 June 2025 — Project opened to the public

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Infrastructure on sphere packings, modular forms, and Fourier analysis

Blueprint refinements: SL's proof of the modular form inequalities

Dabbling in metaprogramming

Groundwork for constructing the magic function

13 June 2025 — Project opened to the public

Growing interest from community contributors

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Infrastructure on sphere packings, modular forms, and Fourier analysis

Blueprint refinements: SL's proof of the modular form inequalities

Dabbling in metaprogramming

Groundwork for constructing the magic function

13 June 2025 — Project opened to the public

Growing interest from community contributors

20 October 2025 — First packathon

Timeline

28 February 2024 — First commit: SH and MV launch project

Drafting the blueprint, working on basic definitions

Late July 2024 — First stable definition of a sphere packing

Infrastructure on sphere packings, modular forms, and Fourier analysis

Blueprint refinements: SL's proof of the modular form inequalities

Dabbling in metaprogramming

Groundwork for constructing the magic function

13 June 2025 — Project opened to the public

Growing interest from community contributors

20 October 2025 — First packathon

Interest from AI companies

Timeline

28 February 2024	First commit: SH and MV launch project
	Drafting the blueprint, working on basic definitions
Late July 2024	First stable definition of a sphere packing
	Infrastructure on sphere packings, modular forms, and Fourier analysis
	Blueprint refinements: SL's proof of the modular form inequalities
	Dabbling in metaprogramming
	Groundwork for constructing the magic function
13 June 2025	Project opened to the public
	Growing interest from community contributors
20 October 2025	First packathon
	Interest from AI companies
February 2026	Project milestone: formal verification

A Wealth of Infrastructure

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

Modular Forms (CB, SL)

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Quasimodular Forms and Inequalities (SL)

A Wealth of Infrastructure

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [3, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Quasimodular Forms and Inequalities (SL)

- Serre derivatives: definitions and API, equivariance and invariance
- Serre derivatives of E_2, E_4, E_6
- Defining F and G, restriction to the imaginary axis, positivity of restrictions
- Modular linear differential equations

Metaprogramming

A `norm_num` extension for $\mathbb{C}$

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta
- Proves `Tendsto (fun z => expr(`f_1 `z, ..., `f_n `z)) l (nhds c)` where `Tendsto` f_i `l (nhds` a_i`)` are known and the expression is continuous

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

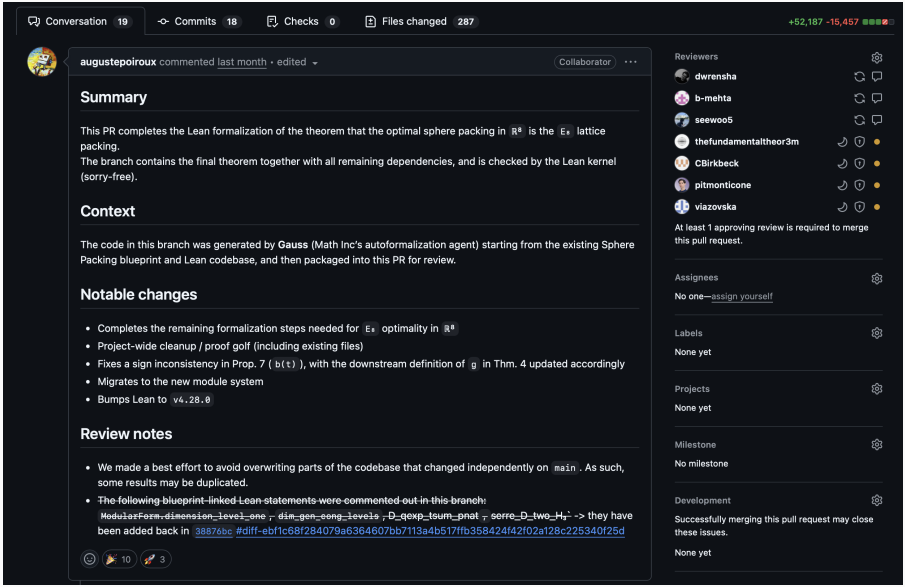
- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta
- Proves `Tendsto (fun z => expr(`f_1 `z, ..., `f_n `z)) l (nhds c)` where `Tendsto` f_i `l (nhds` a_i`)` are known and the expression is continuous

```
-- Two atoms, polynomial
example (h₁ : Tendsto f atTop (nhds 0)) (h₂ :
    Tendsto g atTop (nhds 1)) :
    Tendsto (fun z => f z ^ 2 + f z * g z + g
    z ^ 2) atTop (nhds 1) := by tendsto_cont

-- Unused hypotheses don't interfere
example (h₁ : Tendsto f atTop (nhds 0)) (h₂ :
    Tendsto g atTop (nhds 1))
    (unrelated : Tendsto f atBot (nhds 5)) :
    Tendsto (fun z => f z * g z) atTop (nhds
    0) := by tendsto_cont
```

Formal Verification

Formal Verification



Conversation 19 · Commits 18 · Checks 0 · Files changed 287 · +52,187 -15,457

augustepoiroux commented last month • edited · Collaborator

Summary

This PR completes the Lean formalization of the theorem that the optimal sphere packing in $\mathbb{R}^8$ is the E_8 lattice packing.
The branch contains the final theorem together with all remaining dependencies, and is checked by the Lean kernel (sorry-free).

Context

The code in this branch was generated by **Gauss** (Math Inc's autoformalization agent) starting from the existing Sphere Packing blueprint and Lean codebase, and then packaged into this PR for review.

Notable changes

- Completes the remaining formalization steps needed for E_8 optimality in $\mathbb{R}^8$
- Project-wide cleanup / proof golf (including existing files)
- Fixes a sign inconsistency in Prop. 7 (`b(t)`), with the downstream definition of `g` in Thm. 4 updated accordingly
- Migrates to the new module system
- Bumps Lean to `v4.28.0`

Review notes

- We made a best effort to avoid overwriting parts of the codebase that changed independently on `main` . As such, some results may be duplicated.
- ~~The following blueprint-linked Lean statements were commented out in this branch:~~
  ~~`ModularForm.dimension_level_one`~~ , ~~`dim_gen_cong_levels`~~ , ~~`D_qexp_tsum_pnat`~~ , ~~`serre_D_two_H`~~ -> they have been added back in `3887ebc` #diff-ebf1c68f284079a6364607bb7113a4b517ffb358424f42f02a128c225340f25d

🎉 10 · 🚀 3

Reviewers
dwrensha
b-mehta
seewoo5
thefundamentaltheor3m
CBirkbeck
pitmonticone
viazovska

At least 1 approving review is required to merge this pull request.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

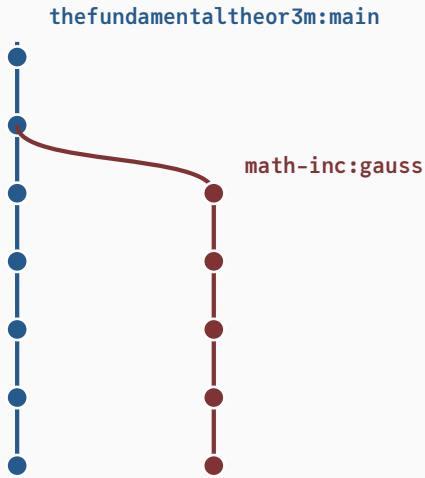
Development
Successfully merging this pull request may close these issues.
None yet

`main` as the Source of Truth

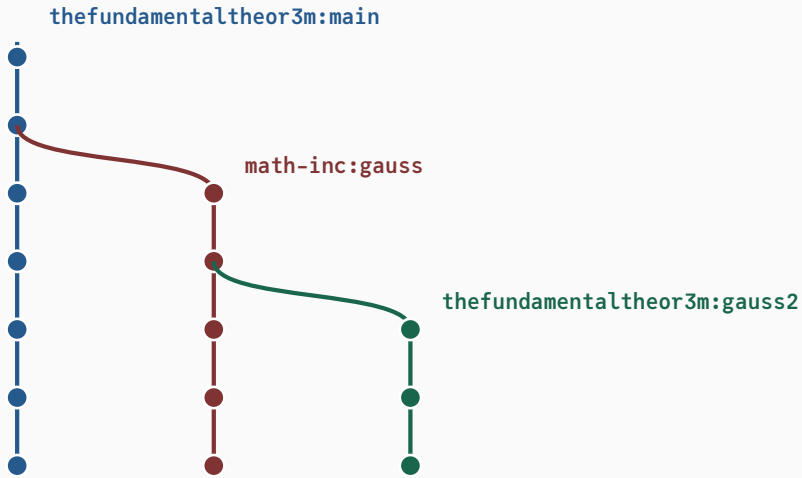
`main` as the Source of Truth

`thefundamentaltheor3m:main`

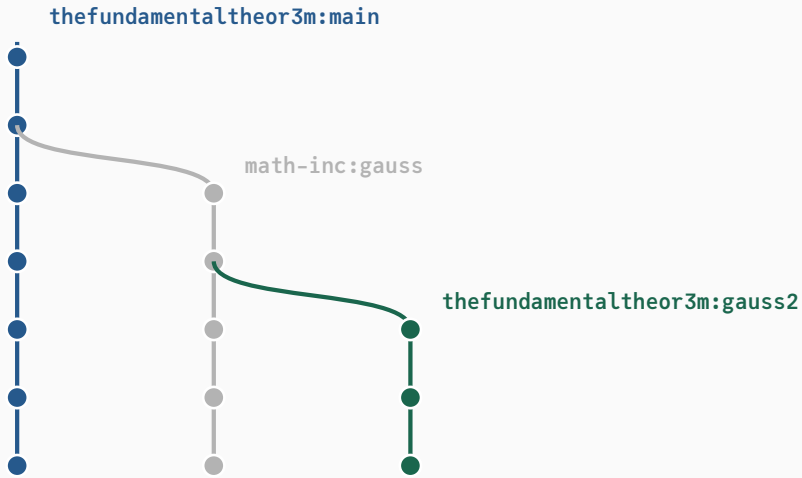
`main` as the Source of Truth



`main` as the Source of Truth

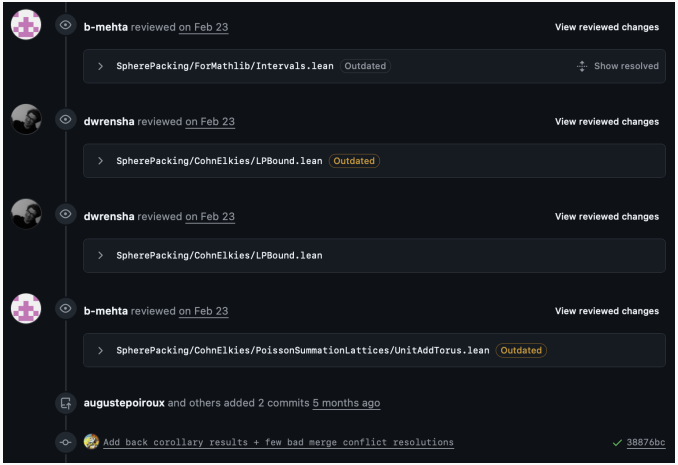


`main` as the Source of Truth



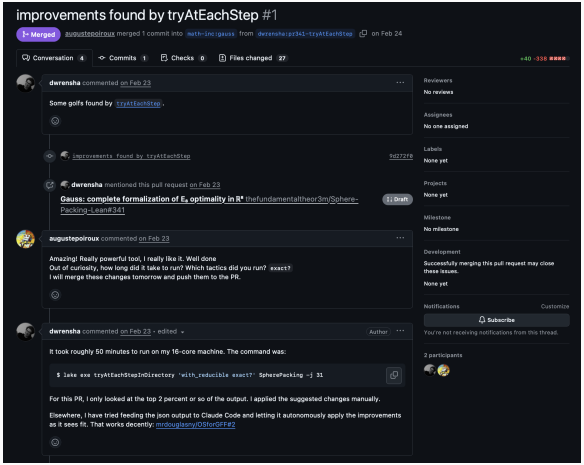
Initial Reviews

Initial Reviews



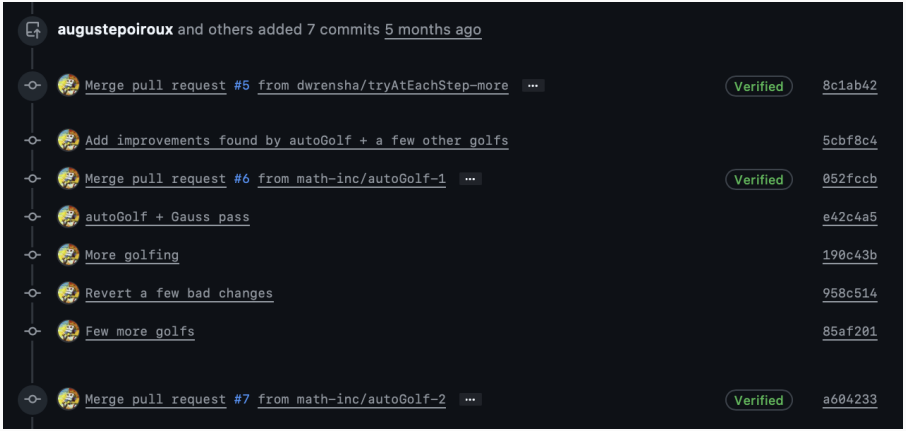
David Renshaw's `TryAtEachStep`

David Renshaw's `TryAtEachStep`

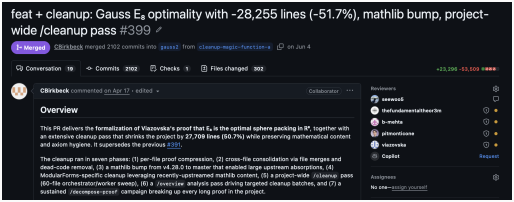


Math, Inc's `AutoGolf`

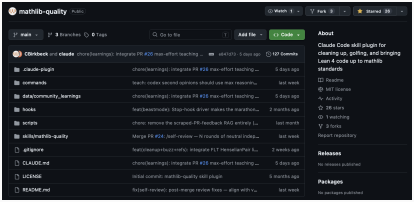
Math, Inc's AutoGolf



The Many Claudes of Christopher Birkbeck



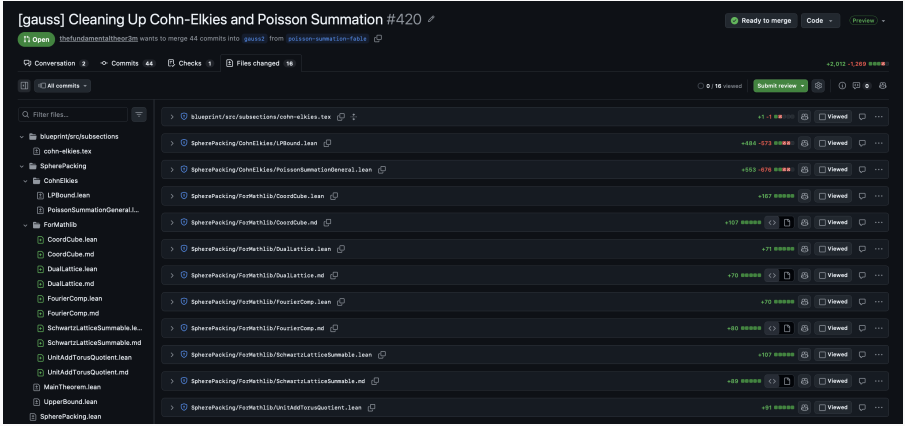
(a) The pull requests



(b) The `mathlib-quality` repository

Targeted Claudeing

Targeted Claudeing



Is it really working?

Is it really working?

At some point, the effectiveness of AI-powered cleanup begins to plateau.

Is it really working?

At some point, the effectiveness of AI-powered cleanup begins to plateau.

The diff is currently better:

Is it really working?

At some point, the effectiveness of AI-powered cleanup begins to plateau.

The diff is currently better:

But lines of code isn't everything. The code quality still needs improvement.

Is it really working?

At some point, the effectiveness of AI-powered cleanup begins to plateau.

The diff is currently better:

But lines of code isn't everything. The code quality still needs improvement.

In the meantime, progress on `main` has become uncomfortably slow: 45 PRs merged between 23 February 2026 and today as opposed to 150 merged between 1 September 2025 and 31 January 2026.

Towards Autocanonisation

Learnings from the Sphere Packing Project

Learnings from the Sphere Packing Project

Our attempt to clean up the sphere packing code demonstrates that AI models currently struggle to canonise an uncanonised verification.

Learnings from the Sphere Packing Project

Our attempt to clean up the sphere packing code demonstrates that AI models currently struggle to canonise an uncanonised verification.

But AI models have already improved by leaps and bounds.

Learnings from the Sphere Packing Project

Our attempt to clean up the sphere packing code demonstrates that AI models currently struggle to canonise an uncanonised verification.

But AI models have already improved by leaps and bounds.

No model today writes code nearly as bad as the original sphere packing Gauss code.

Learnings from the Sphere Packing Project

Our attempt to clean up the sphere packing code demonstrates that AI models currently struggle to canonise an uncanonised verification.

But AI models have already improved by leaps and bounds.

No model today writes code nearly as bad as the original sphere packing Gauss code.

And yet, none have been able to clean up the Gauss code completely.

Learnings from the Sphere Packing Project

Our attempt to clean up the sphere packing code demonstrates that AI models currently struggle to canonise an uncanonised verification.

But AI models have already improved by leaps and bounds.

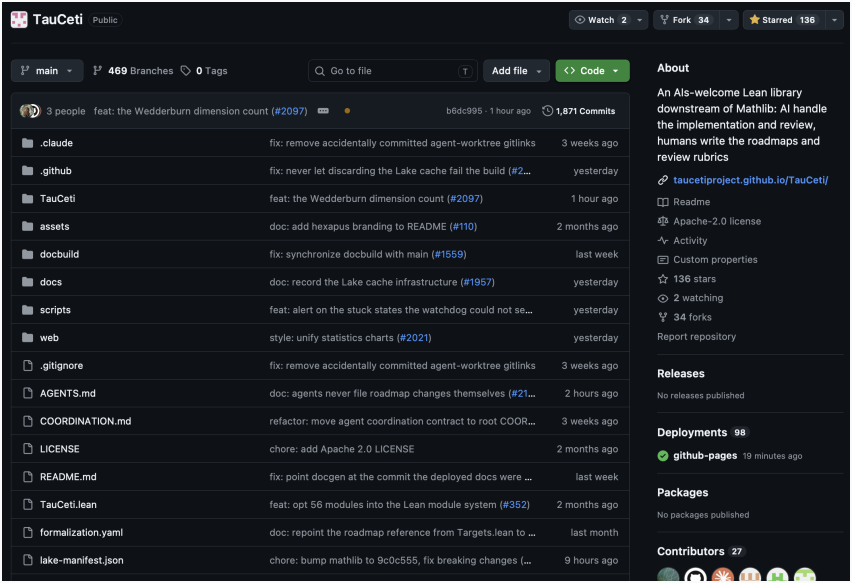
No model today writes code nearly as bad as the original sphere packing Gauss code.

And yet, none have been able to clean up the Gauss code completely.

“It is better to not make a mess than to clean one up.”

Tau Ceti

Tau Ceti


TauCeti
Public
Watch 2
Fork 34
Starred 136
main
469 Branches
0 Tags
Go to file
Add file
Code
3 people
feat: the Wedderburn dimension count (#2097)
b6dc995 · 1 hour ago
1,871 Commits
.claude
fix: remove accidentally committed agent-worktree gitlinks
3 weeks ago
.github
fix: never let discarding the Lake cache fail the build (#2...
yesterday
TauCeti
feat: the Wedderburn dimension count (#2097)
1 hour ago
assets
doc: add hexapus branding to README (#110)
2 months ago
docbuild
fix: synchronize docbuild with main (#1559)
last week
docs
doc: record the Lake cache infrastructure (#1957)
yesterday
scripts
feat: alert on the stuck states the watchdog could not se...
yesterday
web
style: unify statistics charts (#2021)
yesterday
.gitignore
fix: remove accidentally committed agent-worktree gitlinks
3 weeks ago
AGENTS.md
doc: agents never file roadmap changes themselves (#21...
2 hours ago
COORDINATION.md
refactor: move agent coordination contract to root COOR...
3 weeks ago
LICENSE
chore: add Apache 2.0 LICENSE
2 months ago
README.md
fix: point docgen at the commit the deployed docs were ...
last week
TauCeti.lean
feat: opt 56 modules into the Lean module system (#352)
2 months ago
formalization.yaml
doc: repoint the roadmap reference from Targets.lean to ...
last month
lake-manifest.json
chore: bump mathlib to 9c0c555, fix breaking changes (...
9 hours ago
About
An AIs-welcome Lean library downstream of Mathlib: AI handle the implementation and review, humans write the roadmaps and review rubrics
taucetiproject.github.io/TauCeti/
Readme
Apache-2.0 license
Activity
Custom properties
136 stars
2 watching
34 forks
Report repository
Releases
No releases published
Deployments 98
github-pages 19 minutes ago
Packages
No packages published
Contributors 27

Questions?

References i

[1] J. Avigad, K. Donnelly, D. Gray, and P. Raff.
A formally verified proof of the prime number theorem.
ACM Trans. Comput. Logic, 9(1):2–es, Dec. 2007.

[2] K. Buzzard, J. Commelin, and P. Massot.
Formalising perfectoid spaces.
In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 299–312, New York, NY, USA, 2020. Association for Computing Machinery.

[3] H. Cohn and N. Elkies.
New Upper Bounds on Sphere Packings I.
Annals of Mathematics, 157(2):689–714, 2003.

[4] T. M. Community.
The liquid tensor experiment.

[5] M. Eberl.
Nine Chapters of Analytic Number Theory in Isabelle/HOL.
In J. Harrison, J. O'Leary, and A. Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[6] G. Gonthier.
A computer-checked proof of the Four Color Theorem.
Technical report, Inria, Mar. 2023.

[7] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S. O. Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, and L. Théry.
A machine-checked proof of the odd order theorem.
In *Proceedings of the 4th International Conference on Interactive Theorem Proving*, ITP'13, page 163–179, Berlin, Heidelberg, 2013. Springer-Verlag.

[8] T. F. Görbe.
Exceptionally Beautiful Symmetries.
https://tamasgorbe.com/symmetry.

[9] T. Hales.
The formal proof of the kepler conjecture: a critical retrospective, 2024.

[10] T. C. Hales.
Introduction to the Flyspeck Project.
In T. Coquand, H. Lombardi, and M.-F. Roy, editors, *Mathematics, Algorithms, Proofs*, volume 5021 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pages 1–11, Dagstuhl, Germany, 2006. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[11] T. C. Hales.
The jordan curve theorem, formally and informally.
The American Mathematical Monthly, 114(10):882–894, 2007.

[12] J. Harrison.
Formalizing an analytic proof of the prime number theorem.
Journal of Automated Reasoning, 43(3):243–261, Oct 2009.

[13] J. Harrison.
The HOL Light Theory of Euclidean Space.
Journal of Automated Reasoning, 50(2):173–190, Feb 2013.

[14] B. Mehta.
A formal proof of an exponentially better upper bound on Ramsey numbers.
`https://github.com/b-mehta/exponential-ramsey`, 2023.

[15] T. C. Hales et al.
A Formal Proof of the Kepler Conjecture.
Forum of Mathematics, Pi, 5:e2, 2017.

[16] M. S. Viazovska.
The sphere packing problem in dimension 8.
Annals of Mathematics, 185(3):991–1015, 2017.