Formalising modern research mathematics

Kevin Buzzard, Imperial College London

Computer-verified proofs: 48 hours in Rome, 26th January 2024

Before we start

Thank you very much to Oliver Butterley for the invitation, and thanks to all of you for coming!

Before we start

Thank you very much to Oliver Butterley for the invitation, and thanks to all of you for coming!

Plan for the talk:

- A bit of history;
- The rise of mathlib;
- Formalising modern mathematics.

Before we start

Thank you very much to Oliver Butterley for the invitation, and thanks to all of you for coming!

Plan for the talk:

- A bit of history;
- The rise of mathlib;
- Formalising modern mathematics.

In my mind, currently the most exciting thing about this area is that five years ago, the third item seemed like science fiction, but now it is happening.

Interactive theorem provers

An interactive theorem prover (ITP) is a computer programming language which is expressive enough to understand mathematical theorems and proofs.

Interactive theorem provers

An interactive theorem prover (ITP) is a computer programming language which is expressive enough to understand mathematical theorems and proofs.

Note: this is different to a computer algebra system.

Interactive theorem provers

An interactive theorem prover (ITP) is a computer programming language which is expressive enough to understand mathematical theorems and proofs.

Note: this is different to a computer algebra system.

In a computer algebra system, you can *calculate*.

Interactive theorem provers

An interactive theorem prover (ITP) is a computer programming language which is expressive enough to understand mathematical theorems and proofs.

Note: this is different to a computer algebra system.

In a computer algebra system, you can *calculate*.

For example, you can print out the first 1000 prime numbers, or say “let G be the group S_5”.

Interactive theorem provers

An interactive theorem prover (ITP) is a computer programming language which is expressive enough to understand mathematical theorems and proofs.

Note: this is different to a computer algebra system.

In a computer algebra system, you can *calculate*.

For example, you can print out the first 1000 prime numbers, or say "let G be the group S_5".

In an interactive theorem prover, you can *prove*.

Interactive theorem provers

An interactive theorem prover (ITP) is a computer programming language which is expressive enough to understand mathematical theorems and proofs.

Note: this is different to a computer algebra system.

In a computer algebra system, you can *calculate*.

For example, you can print out the first 1000 prime numbers, or say "let G be the group S_5".

In an interactive theorem prover, you can *prove*.

For example, you can prove that there are infinitely many prime numbers, or say "let G be a group".

History: calculating vs proving.

Let's see what computers were doing by the 1970s.

History: calculating vs proving.

Let's see what computers were doing by the 1970s.

Calculating:
The Lehmers and Vandiver proved Fermat's Last Theorem for all $n \leq 2000$ in the 50s.

History: calculating vs proving.

Let's see what computers were doing by the 1970s.

Calculating:
The Lehmers and Vandiver proved Fermat's Last Theorem for all $n \leq 2000$ in the 50s.

In the 60s, Birch and Swinnerton-Dyer computed the number of solutions to various two-variable cubic equations modulo various primes and postulated a conjecture which changed mathematics.

History: calculating vs proving.

Let's see what computers were doing by the 1970s.

Calculating:
The Lehmers and Vandiver proved Fermat's Last Theorem for all $n \leq 2000$ in the 50s.

In the 60s, Birch and Swinnerton-Dyer computed the number of solutions to various two-variable cubic equations modulo various primes and postulated a conjecture which changed mathematics.

Proving:
In the 70s, van Benthem Jutting (a student of de Bruijn) proved that the reals were a complete ordered field.

History: calculating vs proving.

Let's see what computers were doing by the 1970s.

Calculating:
The Lehmers and Vandiver proved Fermat's Last Theorem for all $n \leq 2000$ in the 50s.

In the 60s, Birch and Swinnerton-Dyer computed the number of solutions to various two-variable cubic equations modulo various primes and postulated a conjecture which changed mathematics.

Proving:
In the 70s, van Benthem Jutting (a student of de Bruijn) proved that the reals were a complete ordered field.

You can guess which application caught on amongst mathematicians.

A crucial difference

Computer algebra packages have become very easy for mathematicians to learn and use.

A crucial difference

Computer algebra packages have become very easy for mathematicians to learn and use.

Nowadays there are many mathematicians in a typical mathematics department who have used computer algebra packages or programming languages in their work.

A crucial difference

Computer algebra packages have become very easy for mathematicians to learn and use.

Nowadays there are many mathematicians in a typical mathematics department who have used computer algebra packages or programming languages in their work.

For *decades*, ITPs were very hard for mathematicians to use (and we're trying to fix this with workshops like this one).

A crucial difference

Computer algebra packages have become very easy for mathematicians to learn and use.

Nowadays there are many mathematicians in a typical mathematics department who have used computer algebra packages or programming languages in their work.

For *decades*, ITPs were very hard for mathematicians to use (and we're trying to fix this with workshops like this one).

As a result, ITP growth was mostly in computer science departments.

A crucial difference

Computer algebra packages have become very easy for mathematicians to learn and use.

Nowadays there are many mathematicians in a typical mathematics department who have used computer algebra packages or programming languages in their work.

For *decades*, ITPs were very hard for mathematicians to use (and we're trying to fix this with workshops like this one).

As a result, ITP growth was mostly in computer science departments.

This had a *huge* effect on the kind of mathematics which was being done in them.

ITPs in the 21st century.

Milestone results from the begining of the 21st century:

ITPs in the 21st century.

Milestone results from the begining of the 21st century:

2004: Avigad et al formally proved the prime number theorem.

ITPs in the 21st century.

Milestone results from the begining of the 21st century:

2004: Avigad et al formally proved the prime number theorem.

To the computer scientists: a breakthrough result (discrete and continuous mathematics being handled by the same system).

ITPs in the 21st century.

Milestone results from the begining of the 21st century:

2004: Avigad et al formally proved the prime number theorem.

To the computer scientists: a breakthrough result (discrete and continuous mathematics being handled by the same system).

To the mathematicians: a completely standard 100 year old result which is in an undergraduate or MSc curriculum.

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

The proof involves a gigantic very tedious case check (perfect for a computer).

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

The proof involves a gigantic very tedious case check (perfect for a computer).

2012: A team led by Gonthier formalised a proof of the Feit–Thompson odd-order theorem.

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

The proof involves a gigantic very tedious case check (perfect for a computer).

2012: A team led by Gonthier formalised a proof of the Feit–Thompson odd-order theorem.

The proof is 300 pages of lemmas about finite groups.

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

The proof involves a gigantic very tedious case check (perfect for a computer).

2012: A team led by Gonthier formalised a proof of the Feit–Thompson odd-order theorem.

The proof is 300 pages of lemmas about finite groups.

Still 40 years out of date.

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

The proof involves a gigantic very tedious case check (perfect for a computer).

2012: A team led by Gonthier formalised a proof of the Feit–Thompson odd-order theorem.

The proof is 300 pages of lemmas about finite groups.

Still 40 years out of date.

Arguably “unfashionable mathematics”.

ITPs in the 21st century.

2004: Georges Gonthier formalised a proof of the four colour theorem.

The proof involves a gigantic very tedious case check (perfect for a computer).

2012: A team led by Gonthier formalised a proof of the Feit–Thompson odd-order theorem.

The proof is 300 pages of lemmas about finite groups.

Still 40 years out of date.

Arguably “unfashionable mathematics”.

And these are complex theorems about *simple mathematical objects*.

ITPs in the 21st century.

2017: Hales formalised his proof with Ferguson of the Kepler conjecture (sphere-packing in 3 dimensions).

ITPs in the 21st century.

2017: Hales formalised his proof with Ferguson of the Kepler conjecture (sphere-packing in 3 dimensions).

This was 12 years out of date (because it took Hales' team 12 years to formalise the proof).

ITPs in the 21st century.

2017: Hales formalised his proof with Ferguson of the Kepler conjecture (sphere-packing in 3 dimensions).

This was 12 years out of date (because it took Hales' team 12 years to formalise the proof).

Again, a complex theorem about simple mathematical objects.

What about complex objects?

What about complex objects?

Lean is an ITP, and in 2017 it was only a couple of years old with a small mathematics library (groups, rings, topological spaces, finiteness, rationals and real numbers).

What about complex objects?

Lean is an ITP, and in 2017 it was only a couple of years old with a small mathematics library (groups, rings, topological spaces, finiteness, rationals and real numbers).

In 2017 I started a Lean club at Imperial College London (knowing nothing about ITPs, and having learnt about them by spamming the Lean chat).

What about complex objects?

Lean is an ITP, and in 2017 it was only a couple of years old with a small mathematics library (groups, rings, topological spaces, finiteness, rationals and real numbers).

In 2017 I started a Lean club at Imperial College London (knowing nothing about ITPs, and having learnt about them by spamming the Lean chat).

Some smart 1st year undergraduates came along asking for projects.

What about complex objects?

Lean is an ITP, and in 2017 it was only a couple of years old with a small mathematics library (groups, rings, topological spaces, finiteness, rationals and real numbers).

In 2017 I started a Lean club at Imperial College London (knowing nothing about ITPs, and having learnt about them by spamming the Lean chat).

Some smart 1st year undergraduates came along asking for projects.

I suggested commutative algebra.

What about complex objects?

Lean is an ITP, and in 2017 it was only a couple of years old with a small mathematics library (groups, rings, topological spaces, finiteness, rationals and real numbers).

In 2017 I started a Lean club at Imperial College London (knowing nothing about ITPs, and having learnt about them by spamming the Lean chat).

Some smart 1st year undergraduates came along asking for projects.

I suggested commutative algebra. Localisation of rings was completed two weeks later by Kenny Lau.

What about complex objects?

Lean is an ITP, and in 2017 it was only a couple of years old with a small mathematics library (groups, rings, topological spaces, finiteness, rationals and real numbers).

In 2017 I started a Lean club at Imperial College London (knowing nothing about ITPs, and having learnt about them by spamming the Lean chat).

Some smart 1st year undergraduates came along asking for projects.

I suggested commutative algebra. Localisation of rings was completed two weeks later by Kenny Lau.

So I suggested "definition of a scheme", which took us two months.

Schemes

Once we had a working definition of scheme, the undergraduates started formalising basic exercises in Hartshorne's book on algebraic geometry.

Schemes

Once we had a working definition of scheme, the undergraduates started formalising basic exercises in Hartshorne's book on algebraic geometry.

I had got 1st year undergraduates doing masters level mathematics!

Schemes

Once we had a working definition of scheme, the undergraduates started formalising basic exercises in Hartshorne's book on algebraic geometry.

I had got 1st year undergraduates doing masters level mathematics!

And I started asking how schemes were formalised in the other systems.

Schemes

Once we had a working definition of scheme, the undergraduates started formalising basic exercises in Hartshorne's book on algebraic geometry.

I had got 1st year undergraduates doing masters level mathematics!

And I started asking how schemes were formalised in the other systems.

Turns out that they had never been done.

Schemes

Once we had a working definition of scheme, the undergraduates started formalising basic exercises in Hartshorne's book on algebraic geometry.

I had got 1st year undergraduates doing masters level mathematics!

And I started asking how schemes were formalised in the other systems.

Turns out that they had never been done.

Computer scientists were giving me suggestions about where to *publish the work!*

Schemes

Once we had a working definition of scheme, the undergraduates started formalising basic exercises in Hartshorne's book on algebraic geometry.

I had got 1st year undergraduates doing masters level mathematics!

And I started asking how schemes were formalised in the other systems.

Turns out that they had never been done.

Computer scientists were giving me suggestions about where to *publish the work!*

But this was not my "research" – this was just a fun project!

Schemes

I told some mathematicians in my department that I'd taught a computer to understand the sentence "let X be a scheme".

Schemes

I told some mathematicians in my department that I'd taught a computer to understand the sentence "let X be a scheme".

They were super-unimpressed!

Schemes

I told some mathematicians in my department that I'd taught a computer to understand the sentence "let X be a scheme".

They were super-unimpressed!

"I knew what a scheme was when I was a PhD student Kevin!"

Schemes

I told some mathematicians in my department that I'd taught a computer to understand the sentence "let X be a scheme".

They were super-unimpressed!

"I knew what a scheme was when I was a PhD student Kevin!"

Schemes were not complex enough.

Perfectoid spaces

At the end of 2017 Patrick Massot and I (independently!) decided to formalise the definition of a perfectoid space, in an attempt to solve the problem of mathematicians not being remotely interested in this area.

Perfectoid spaces

At the end of 2017 Patrick Massot and I (independently!) decided to formalise the definition of a perfectoid space, in an attempt to solve the problem of mathematicians not being remotely interested in this area.

It was at that time an open secret that Scholze was going to get the Fields Medal in 2018 for his work on perfectoid spaces.

Perfectoid spaces

At the end of 2017 Patrick Massot and I (independently!) decided to formalise the definition of a perfectoid space, in an attempt to solve the problem of mathematicians not being remotely interested in this area.

It was at that time an open secret that Scholze was going to get the Fields Medal in 2018 for his work on perfectoid spaces.

We decided to go for it.

Perfectoid spaces

Perfectoid spaces had applications in arithmetic geometry, commutative algebra and the Langlands Program.

Perfectoid spaces

Perfectoid spaces had applications in arithmetic geometry, commutative algebra and the Langlands Program.

We proved *none* of these applications – they were completely unfeasible.

Perfectoid spaces

Perfectoid spaces had applications in arithmetic geometry, commutative algebra and the Langlands Program.

We proved *none* of these applications – they were completely unfeasible.

But by this point Lean's mathematics library `mathlib` had enough background material in algebra and topology to make the task of formalising the *definition* feasible.

Perfectoid spaces

Perfectoid spaces had applications in arithmetic geometry, commutative algebra and the Langlands Program.

We proved *none* of these applications – they were completely unfeasible.

But by this point Lean's mathematics library `mathlib` had enough background material in algebra and topology to make the task of formalising the *definition* feasible.

With Johan Commelin (who heard about the schemes work and got interested), about 15000 lines of code and 18 months later, we had built up enough of the theory of topological rings, valuations, sheaves etc, to formalise the *definition*.

Perfectoid spaces

Perfectoid spaces had applications in arithmetic geometry, commutative algebra and the Langlands Program.

We proved *none* of these applications – they were completely unfeasible.

But by this point Lean's mathematics library `mathlib` had enough background material in algebra and topology to make the task of formalising the *definition* feasible.

With Johan Commelin (who heard about the schemes work and got interested), about 15000 lines of code and 18 months later, we had built up enough of the theory of topological rings, valuations, sheaves etc, to formalise the *definition*.

It was suggested we prove a theorem about perfectoid spaces.

Perfectoid spaces

Perfectoid spaces had applications in arithmetic geometry, commutative algebra and the Langlands Program.

We proved *none* of these applications – they were completely unfeasible.

But by this point Lean's mathematics library `mathlib` had enough background material in algebra and topology to make the task of formalising the *definition* feasible.

With Johan Commelin (who heard about the schemes work and got interested), about 15000 lines of code and 18 months later, we had built up enough of the theory of topological rings, valuations, sheaves etc, to formalise the *definition*.

It was suggested we prove a theorem about perfectoid spaces. So we proved that the empty set was a perfectoid space.

Perfectoid spaces

The response to our work was very surprising (to me).

Perfectoid spaces

The response to our work was very surprising (to me).

The formalisation was a relatively straightforward exercise, and a big publicity stunt.

Perfectoid spaces

The response to our work was very surprising (to me).

The formalisation was a relatively straightforward exercise, and a big publicity stunt.

But all of a sudden mathematicians were interested!

Perfectoid spaces

The response to our work was very surprising (to me).

The formalisation was a relatively straightforward exercise, and a big publicity stunt.

But all of a sudden mathematicians were interested!

A bunch of Italian number theorists showed up on the Lean chat :-)

Ellenberg–Gijswijt

In 2017 Ellenberg and Gijswijt proved a conjecture in combinatorics called the Cap Set conjecture.

Ellenberg–Gijswijt

In 2017 Ellenberg and Gijswijt proved a conjecture in combinatorics called the Cap Set conjecture.

Their work was published in the Annals of Mathematics, and had very few prerequisites.

Ellenberg–Gijswijt

In 2017 Ellenberg and Gijswijt proved a conjecture in combinatorics called the Cap Set conjecture.

Their work was published in the Annals of Mathematics, and had very few prerequisites.

Again using `mathlib`, Dahmen, Hoelzl and Lewis completely formalised the proof in 2019.

Ellenberg–Gijswijt

In 2017 Ellenberg and Gijswijt proved a conjecture in combinatorics called the Cap Set conjecture.

Their work was published in the Annals of Mathematics, and had very few prerequisites.

Again using `mathlib`, Dahmen, Hoelzl and Lewis completely formalised the proof in 2019.

Two years between proof and formalisation, and it was an Annals paper!

Ellenberg–Gijswijt

In 2017 Ellenberg and Gijswijt proved a conjecture in combinatorics called the Cap Set conjecture.

Their work was published in the Annals of Mathematics, and had very few prerequisites.

Again using `mathlib`, Dahmen, Hoelzl and Lewis completely formalised the proof in 2019.

Two years between proof and formalisation, and it was an Annals paper!

The paper was four pages long :-)

Scholze

In 2020 the community got our first real challenge.

Scholze

In 2020 the community got our first real challenge.

The area had plenty of complex proofs about simple objects, and one simple proof about a very complex object.

Scholze

In 2020 the community got our first real challenge.

The area had plenty of complex proofs about simple objects, and one simple proof about a very complex object.

Then Scholze challenged the community to prove a complex theorem about complex objects.

Scholze

In 2020 the community got our first real challenge.

The area had plenty of complex proofs about simple objects, and one simple proof about a very complex object.

Then Scholze challenged the community to prove a complex theorem about complex objects.

The challenge: prove the “fundamental theorem of liquid vector spaces”.

Scholze

In 2020 the community got our first real challenge.

The area had plenty of complex proofs about simple objects, and one simple proof about a very complex object.

Then Scholze challenged the community to prove a complex theorem about complex objects.

The challenge: prove the “fundamental theorem of liquid vector spaces”.

This was a theorem of Clausen and Scholze, announced in 2019 (and still not published AFAIK?)

Scholze

At that time, Commelin had no tenure position, so taking on this project was a huge gamble.

Scholze

At that time, Commelin had no tenure position, so taking on this project was a huge gamble.

Instead of proving new theorems, he was going to check someone else's theorem in Lean.

Scholze

At that time, Commelin had no tenure position, so taking on this project was a huge gamble.

Instead of proving new theorems, he was going to check someone else's theorem in Lean.

But, because of all these projects, and lots of work by undergraduates and Masters students and PhD students, and good organisation and planning (by both mathematicians and computer scientists), `mathlib` was *getting quite good*.

Scholze

At that time, Commelin had no tenure position, so taking on this project was a huge gamble.

Instead of proving new theorems, he was going to check someone else's theorem in Lean.

But, because of all these projects, and lots of work by undergraduates and Masters students and PhD students, and good organisation and planning (by both mathematicians and computer scientists), `mathlib` was *getting quite good*.

The mathematicians were proving the theorems, but the computer scientists were *absolutely essential*, working behind the scenes to ensure that the wheels didn't fall off.

Scholze

At that time, Commelin had no tenure position, so taking on this project was a huge gamble.

Instead of proving new theorems, he was going to check someone else's theorem in Lean.

But, because of all these projects, and lots of work by undergraduates and Masters students and PhD students, and good organisation and planning (by both mathematicians and computer scientists), `mathlib` was *getting quite good*.

The mathematicians were proving the theorems, but the computer scientists were *absolutely essential*, working behind the scenes to ensure that the wheels didn't fall off.

Commelin decided to gamble.

The Liquid Tensor Experiment

The Liquid Tensor Experiment's goal was to prove two theorems.

The Liquid Tensor Experiment

The Liquid Tensor Experiment's goal was to prove two theorems.

1. A technical lemma about double complexes of pseudo-normed abelian groups with a completely elementary (but very messy and long) statement and a ten page proof.

The Liquid Tensor Experiment

The Liquid Tensor Experiment's goal was to prove two theorems.

1. A technical lemma about double complexes of pseudo-normed abelian groups with a completely elementary (but very messy and long) statement and a ten page proof.
2. A corollary (with a 5 line proof in the paper) about the vanishing of higher Ext groups in a certain abelian category of "solid abelian groups".

The Liquid Tensor Experiment

The Liquid Tensor Experiment's goal was to prove two theorems.

1. A technical lemma about double complexes of pseudo-normed abelian groups with a completely elementary (but very messy and long) statement and a ten page proof.
2. A corollary (with a 5 line proof in the paper) about the vanishing of higher Ext groups in a certain abelian category of "solid abelian groups".

At that time, part 2 was the holy grail: a complex theorem about complex objects.

The Liquid Tensor Experiment

The Liquid Tensor Experiment's goal was to prove two theorems.

1. A technical lemma about double complexes of pseudo-normed abelian groups with a completely elementary (but very messy and long) statement and a ten page proof.
2. A corollary (with a 5 line proof in the paper) about the vanishing of higher Ext groups in a certain abelian category of "solid abelian groups".

At that time, part 2 was the holy grail: a complex theorem about complex objects.

Commelin, Topaz, myself, Nuccio, and many others got to work.

The Liquid Tensor Experiment

How does one manage a large multi-author formalisation project?

The Liquid Tensor Experiment

How does one manage a large multi-author formalisation project?

Commelin and Massot created a “formal blueprint”.

The Liquid Tensor Experiment

How does one manage a large multi-author formalisation project?

Commelin and Massot created a “formal blueprint”.

If I have internet, let’s take a look at it.

LaTeX

Blueprint for the Liquid Tensor Experiment

error term a.

As preparation for the proof, we have the following results.

Lemma 1.6.5 (Gordan's lemma)✓

Let Λ be a finite free abelian group, and let $\lambda_1, \ldots, \lambda_m \in \Lambda$ be elements. Let $M \subset \mathrm{Hom}(\Lambda, \mathbb{Z})$ be the submonoid $\{x \mid x(\lambda_i) \geq 0 \text{ for all } i = 1, \ldots, m\}$. Then M is finitely generated as monoid.

Proof ▾

This is a standard result. We omit the proof here. It is done in Lean.

Lemma 1.6.6 ✓

Let Λ be a finite free abelian group, let N be a positive integer, and let $\lambda_1, \ldots, \lambda_m \in \Lambda$ be elements. Then there is a finite subset $A \subset \Lambda^\vee$ such that for all $x \in \Lambda^\vee = \mathrm{Hom}(\Lambda, \mathbb{Z})$ there is some $x' \in A$ such that $x - x' \in N\Lambda^\vee$ and for all $i = 1, \ldots, m$, the numbers $x'(\lambda_i)$ and $(x - x')(\lambda_i)$ have the same sign, i.e. are both nonnegative or both nonpositive.

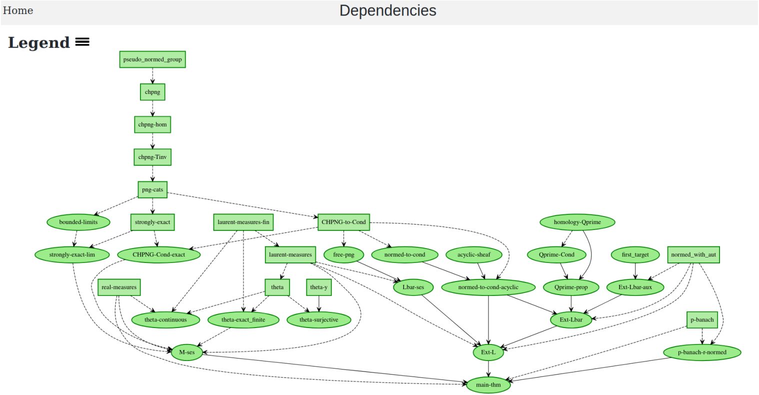
Proof ▾

It suffices to prove the statement for all x such that $\lambda_i(x) \geq 0$ for all i; indeed, applying this variant to all $\pm\lambda_i$, one gets the full statement.

Thus, consider the submonoid $\Lambda_+^\vee \subset \Lambda^\vee$ of all x that pair nonnegatively with all λ_i. This is a finitely generated monoid by Lemma 1.6.5; let $y_1, \ldots, y_M$ be a set of generators. Then we can take for A all sums $n_1 y_1 + \ldots + n_M y_M$ where all $n_j \in \{0, \ldots, N-1\}$.

Lemma 1.6.7 ✓

Graph


Home
Dependencies
Legend
pseudo_normed_group
chpng
chpng-hom
chpng-Tinv
png-cats
bounded-limits
strongly-exact
laurent-measures-fin
CHPNG-to-Cond
homology-Qprime
strongly-exact-lim
CHPNG-Cond-exact
laurent-measures
free-png
normed-to-cond
acyclic-sheaf
Qprime-Cond
first_target
normed_with_aut
real-measures
theta
theta-y
Lbar-ses
normed-to-cond-acyclic
Qprime-prop
Ext-Lbar-aux
theta-continuous
theta-exact_finite
theta-surjective
Ext-Lbar
p-banach
M-ses
Ext-L
p-banach-r-normed
main-thm

Project management

The idea: the blueprint contains a *detailed mathematical argument*, with no phrases like “Now we can easily see that…” or “After some work, we deduce that…” or “It is well-known to the experts that…”.

Project management

The idea: the blueprint contains a *detailed mathematical argument*, with no phrases like "Now we can easily see that…" or "After some work, we deduce that…" or "It is well-known to the experts that…".

The proof is broken up into a sequence of small lemmas, each of which is represented by a node on the blueprint graph.

Project management

The idea: the blueprint contains a *detailed mathematical argument*, with no phrases like "Now we can easily see that…" or "After some work, we deduce that…" or "It is well-known to the experts that…".

The proof is broken up into a sequence of small lemmas, each of which is represented by a node on the blueprint graph.

If a node is blue, this means "we need the Lean code for this proof."

Project management

The idea: the blueprint contains a *detailed mathematical argument*, with no phrases like "Now we can easily see that..." or "After some work, we deduce that..." or "It is well-known to the experts that...".

The proof is broken up into a sequence of small lemmas, each of which is represented by a node on the blueprint graph.

If a node is blue, this means "we need the Lean code for this proof."

If it's green, this means the Lean code is already written.

Project management

This technique means that *many people* can get involved.

Project management

This technique means that *many people* can get involved.

If you are a mathematician who knows *nothing* about liquid vector spaces or solid abelian groups, you can still help!

Project management

This technique means that *many people* can get involved.

If you are a mathematician who knows *nothing* about liquid vector spaces or solid abelian groups, you can still help!

You just find a node which corresponds to a mathematical argument which you understand.

Project management

This technique means that *many people* can get involved.

If you are a mathematician who knows *nothing* about liquid vector spaces or solid abelian groups, you can still help!

You just find a node which corresponds to a mathematical argument which you understand.

Key fact: The experts do not need to check your proof – the computer checks it for you.

The Liquid Tensor Experiment

Six months after Scholze had issued the challenge, the Lean community had formalised a proof of the technical lemma.

The Liquid Tensor Experiment

Six months after Scholze had issued the challenge, the Lean community had formalised a proof of the technical lemma.

Scholze wrote a blog post saying that he was amazed.

The Liquid Tensor Experiment

Six months after Scholze had issued the challenge, the Lean community had formalised a proof of the technical lemma.

Scholze wrote a blog post saying that he was amazed.

The next thing I knew, Nature was on the phone asking what the heck was going on.

The Liquid Tensor Experiment

Six months after Scholze had issued the challenge, the Lean community had formalised a proof of the technical lemma.

Scholze wrote a blog post saying that he was amazed.

The next thing I knew, Nature was on the phone asking what the heck was going on.

But the project was not finished: we still had to formalise the five remaining lines in the proof.

The final five lines

Those five lines assumed a huge amount of homological algebra, derived functors, a formula for the explicit computation of an Ext group, projective objects and so on.

The final five lines

Those five lines assumed a huge amount of homological algebra, derived functors, a formula for the explicit computation of an Ext group, projective objects and so on.

The five lines (containing many “well-known” facts) took 13 months to formalise.

The final five lines

Those five lines assumed a huge amount of homological algebra, derived functors, a formula for the explicit computation of an Ext group, projective objects and so on.

The five lines (containing many “well-known” facts) took 13 months to formalise.

This experience taught the community *many* things.

The art of formalisation

One thing which really hit home for me: this showed us that *formalisation of mathematical definitions is an art*.

The art of formalisation

One thing which really hit home for me: this showed us that *formalisation of mathematical definitions is an art*.

(And this is why I believe that it will be a long time before AI catches up with us).

The art of formalisation

One thing which really hit home for me: this showed us that *formalisation of mathematical definitions is an art*.

(And this is why I believe that it will be a long time before AI catches up with us).

Q: What is *a practical way* to formalise the basic theory of homological algebra in a general abelian category, in Lean's dependent type theory?

The art of formalisation

One thing which really hit home for me: this showed us that *formalisation of mathematical definitions is an art*.

(And this is why I believe that it will be a long time before AI catches up with us).

Q: What is *a practical way* to formalise the basic theory of homological algebra in a general abelian category, in Lean's dependent type theory?

In 2020, nobody in the world knew the answer.

The art of formalisation

One thing which really hit home for me: this showed us that *formalisation of mathematical definitions is an art*.

(And this is why I believe that it will be a long time before AI catches up with us).

Q: What is *a practical way* to formalise the basic theory of homological algebra in a general abelian category, in Lean's dependent type theory?

In 2020, nobody in the world knew the answer.

This is *not an easy question*.

The art of formalisation

One thing which really hit home for me: this showed us that *formalisation of mathematical definitions is an art*.

(And this is why I believe that it will be a long time before AI catches up with us).

Q: What is *a practical way* to formalise the basic theory of homological algebra in a general abelian category, in Lean's dependent type theory?

In 2020, nobody in the world knew the answer.

This is *not an easy question*.

In fact, "how to teach the computer the objects used in modern mathematics" is an *active research area*.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The first answer we tried was the naive one: kernel of d over image of d.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The first answer we tried was the naive one: kernel of d over image of d.

This was what we used in the Liquid Tensor Experiment.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The first answer we tried was the naive one: kernel of d over image of d.

This was what we used in the Liquid Tensor Experiment.

It was hard to work with, and didn't make it into `mathlib`.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The first answer we tried was the naive one: kernel of d over image of d.

This was what we used in the Liquid Tensor Experiment.

It was hard to work with, and didn't make it into `mathlib`.

It was *extremely messy* to prove that homology of the opposite was the opposite of the homology in this generality.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The first answer we tried was the naive one: kernel of d over image of d.

This was what we used in the Liquid Tensor Experiment.

It was hard to work with, and didn't make it into `mathlib`.

It was *extremely messy* to prove that homology of the opposite was the opposite of the homology in this generality.

The homology of $A \to B \to C$ is both a limit and a colimit.

The art of formalisation

In 2024, we now have a good answer to the question of how to formalise homological algebra (thanks to Joel Riou).

But it was not the first answer that we tried.

The first answer we tried was the naive one: kernel of d over image of d.

This was what we used in the Liquid Tensor Experiment.

It was hard to work with, and didn't make it into `mathlib`.

It was *extremely messy* to prove that homology of the opposite was the opposite of the homology in this generality.

The homology of $A \to B \to C$ is both a limit and a colimit.

But the experience taught us what we *should* have done.

Homology of a complex

Joel Riou re-implemented the definition of homology of a complex as a “package” with a beautiful duality.

Homology of a complex

Joel Riou re-implemented the definition of homology of a complex as a “package” with a beautiful duality.

This version made it into `mathlib`.

Homology of a complex

Joel Riou re-implemented the definition of homology of a complex as a “package” with a beautiful duality.

This version made it into `mathlib`.

And now anyone else who wants homology, for any project, can use `mathlib`’s version here.

Homology of a complex

Joel Riou re-implemented the definition of homology of a complex as a “package” with a beautiful duality.

This version made it into `mathlib`.

And now anyone else who wants homology, for any project, can use `mathlib`’s version here.

And we know it is usable.

Analytic number theory

The Liquid Tensor Experiment was a huge multi-author project which took many many person-years, and had huge benefits for `mathlib`.

Analytic number theory

The Liquid Tensor Experiment was a huge multi-author project which took many many person-years, and had huge benefits for `mathlib`.

But in analytic number theory there was another revolution going on.

Analytic number theory

The Liquid Tensor Experiment was a huge multi-author project which took many many person-years, and had huge benefits for `mathlib`.

But in analytic number theory there was another revolution going on.

In 2021 Tom Bloom proved an old conjecture of Erdős and Davenport on unit fractions.

Analytic number theory

The Liquid Tensor Experiment was a huge multi-author project which took many many person-years, and had huge benefits for `mathlib`.

But in analytic number theory there was another revolution going on.

In 2021 Tom Bloom proved an old conjecture of Erdős and Davenport on unit fractions.

The proof uses an instance of the Hardy–Littlewood circle method and some delicate estimates.

Analytic number theory

The Liquid Tensor Experiment was a huge multi-author project which took many many person-years, and had huge benefits for `mathlib`.

But in analytic number theory there was another revolution going on.

In 2021 Tom Bloom proved an old conjecture of Erdős and Davenport on unit fractions.

The proof uses an instance of the Hardy–Littlewood circle method and some delicate estimates.

6 months later Mehta had taught Bloom how to use Lean, and the pair of them had formalised the entire proof *before Bloom had received a referee's report*.

Analytic number theory

The Liquid Tensor Experiment was a huge multi-author project which took many many person-years, and had huge benefits for `mathlib`.

But in analytic number theory there was another revolution going on.

In 2021 Tom Bloom proved an old conjecture of Erdős and Davenport on unit fractions.

The proof uses an instance of the Hardy–Littlewood circle method and some delicate estimates.

6 months later Mehta had taught Bloom how to use Lean, and the pair of them had formalised the entire proof *before Bloom had received a referee's report*.

Mehta wrote an appendix to the paper explaining the work.

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

This result was featured in Quanta's "2023 breakthroughs in mathematics" video.

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

This result was featured in Quanta's "2023 breakthroughs in mathematics" video.

The paper was 55 pages and the arguments took the authors 5 years to come up with.

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

This result was featured in Quanta's "2023 breakthroughs in mathematics" video.

The paper was 55 pages and the arguments took the authors 5 years to come up with.

Mehta formalised the entire proof, by himself, in 5 months.

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

This result was featured in Quanta's "2023 breakthroughs in mathematics" video.

The paper was 55 pages and the arguments took the authors 5 years to come up with.

Mehta formalised the entire proof, by himself, in 5 months.

Again before the paper had been refereed.

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

This result was featured in Quanta's "2023 breakthroughs in mathematics" video.

The paper was 55 pages and the arguments took the authors 5 years to come up with.

Mehta formalised the entire proof, by himself, in 5 months.

Again before the paper had been refereed.

This now really is modern mathematics being formalised "in real time".

Combinatorics

In 2023 Campos, Griffiths, Morris and Sahasrabudhe announced breakthrough new bounds in Ramsey numbers (beating an 89-year-old theorem of Erdős).

This result was featured in Quanta's "2023 breakthroughs in mathematics" video.

The paper was 55 pages and the arguments took the authors 5 years to come up with.

Mehta formalised the entire proof, by himself, in 5 months.

Again before the paper had been refereed.

This now really is modern mathematics being formalised "in real time".

But the real bombshells were yet to come.

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

He wrote a short paper on the topic and uploaded it to ArXiv.

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

He wrote a short paper on the topic and uploaded it to ArXiv.

He then announced that he would learn Lean by formalising his own result in Lean.

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

He wrote a short paper on the topic and uploaded it to ArXiv.

He then announced that he would learn Lean by formalising his own result in Lean.

This was a single-author project.

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

He wrote a short paper on the topic and uploaded it to ArXiv.

He then announced that he would learn Lean by formalising his own result in Lean.

This was a single-author project.

A few weeks later Lean pointed out to him that at some point in the argument he had divided by zero :-)

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

He wrote a short paper on the topic and uploaded it to ArXiv.

He then announced that he would learn Lean by formalising his own result in Lean.

This was a single-author project.

A few weeks later Lean pointed out to him that at some point in the argument he had divided by zero :-)

It was not difficult to fix the problem.

A lemma of Tao

In October 2023, Terry Tao proved an inequality inspired by a question on MathOverflow.

He wrote a short paper on the topic and uploaded it to ArXiv.

He then announced that he would learn Lean by formalising his own result in Lean.

This was a single-author project.

A few weeks later Lean pointed out to him that at some point in the argument he had divided by zero :-)

It was not difficult to fix the problem.

You can read about Tao's experience on his blog.

The polynomial Freiman–Ruzsa conjecture

In November 2023 Gowers, Green, Manners and Tao announced a proof of the polynomial Freiman–Ruzsa conjecture.

The polynomial Freiman–Ruzsa conjecture

In November 2023 Gowers, Green, Manners and Tao announced a proof of the polynomial Freiman–Ruzsa conjecture.

This is an important conjecture in additive combinatorics.

The polynomial Freiman–Ruzsa conjecture

In November 2023 Gowers, Green, Manners and Tao announced a proof of the polynomial Freiman–Ruzsa conjecture.

This is an important conjecture in additive combinatorics.

Four days later Tao announced that he wanted to lead a multi-author formalisation of the 33 page paper.

The polynomial Freiman–Ruzsa conjecture

In November 2023 Gowers, Green, Manners and Tao announced a proof of the polynomial Freiman–Ruzsa conjecture.

This is an important conjecture in additive combinatorics.

Four days later Tao announced that he wanted to lead a multi-author formalisation of the 33 page paper.

Tao used Massot's blueprint software, and cut and pasted much of the paper into his blueprint.

The polynomial Freiman–Ruzsa conjecture

In November 2023 Gowers, Green, Manners and Tao announced a proof of the polynomial Freiman–Ruzsa conjecture.

This is an important conjecture in additive combinatorics.

Four days later Tao announced that he wanted to lead a multi-author formalisation of the 33 page paper.

Tao used Massot's blueprint software, and cut and pasted much of the paper into his blueprint.

25 people joined in and *three weeks later* the result was completely formalised.

The polynomial Freiman–Ruzsa conjecture

In November 2023 Gowers, Green, Manners and Tao announced a proof of the polynomial Freiman–Ruzsa conjecture.

This is an important conjecture in additive combinatorics.

Four days later Tao announced that he wanted to lead a multi-author formalisation of the 33 page paper.

Tao used Massot's blueprint software, and cut and pasted much of the paper into his blueprint.

25 people joined in and *three weeks later* the result was completely formalised.

The formalisation was completed *before the paper was even submitted*.

Conclusions

What do we learn from these stories?

Conclusions

What do we learn from these stories?

- Formalisation of *some* modern mathematics is possible in real time.

Conclusions

What do we learn from these stories?

- Formalisation of *some* modern mathematics is possible in real time.
- If the prerequisites are in `mathlib` then the project is feasible.

Conclusions

What do we learn from these stories?

- Formalisation of *some* modern mathematics is possible in real time.
- If the prerequisites are in `mathlib` then the project is feasible.
- If the prerequisites are *not* in `mathlib`, then you can formalise your result anyway, and *make* `mathlib` *better*.

Conclusions

What do we learn from these stories?

- Formalisation of *some* modern mathematics is possible in real time.
- If the prerequisites are in `mathlib` then the project is feasible.
- If the prerequisites are *not* in `mathlib`, then you can formalise your result anyway, and *make* `mathlib` *better*.
- For certain topics (parts of analytic number theory, parts of additive combinatorics), the prerequisites are now often there.

Conclusions

More lessons:

- The tooling is there to make multi-author projects feasible (and easy!)

Conclusions

More lessons:

- The tooling is there to make multi-author projects feasible (and easy!)
- *A multi-author project is really good fun*.

Conclusions

More lessons:

- The tooling is there to make multi-author projects feasible (and easy!)
- *A multi-author project is really good fun*.
- Your line manager might be confused about your work, which might be published in a journal they've never heard of.

Conclusions

More lessons:

- The tooling is there to make multi-author projects feasible (and easy!)
- *A multi-author project is really good fun*.
- Your line manager might be confused about your work, which might be published in a journal they've never heard of.
- `mathlib` is becoming *enormously powerful* and it is getting better at an *extremely* fast rate.

Fermat's Last Theorem

My line manager was very confused about my work in this area, for some time.

Fermat’s Last Theorem

My line manager was very confused about my work in this area, for some time.

But then I brought in a seven figure grant to formalise a proof of Fermat’s Last Theorem in Lean, and they came around to the idea.

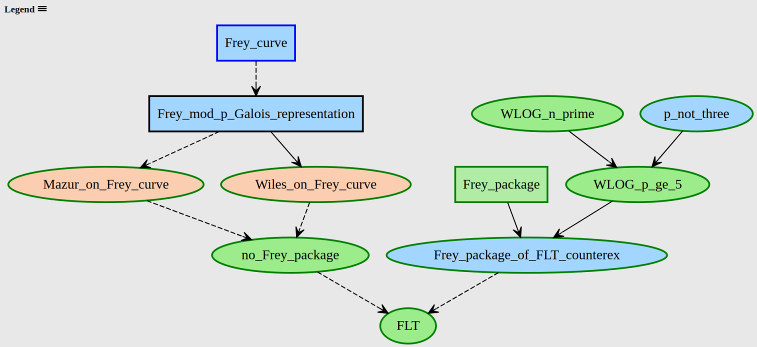
Fermat's Last Theorem

My line manager was very confused about my work in this area, for some time.

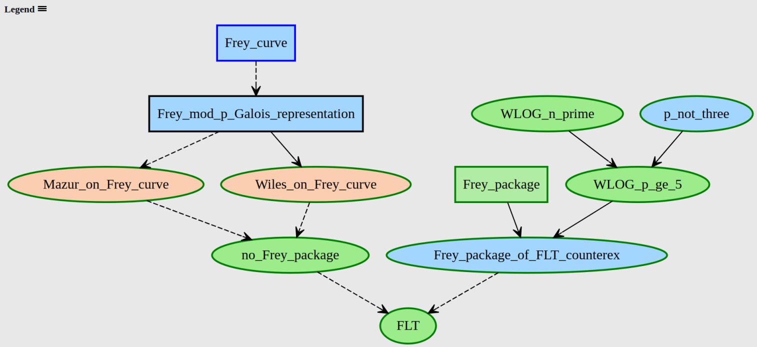
But then I brought in a seven figure grant to formalise a proof of Fermat's Last Theorem in Lean, and they came around to the idea.

Here is the current state of my (secret) blueprint:

FLT blueprint graph

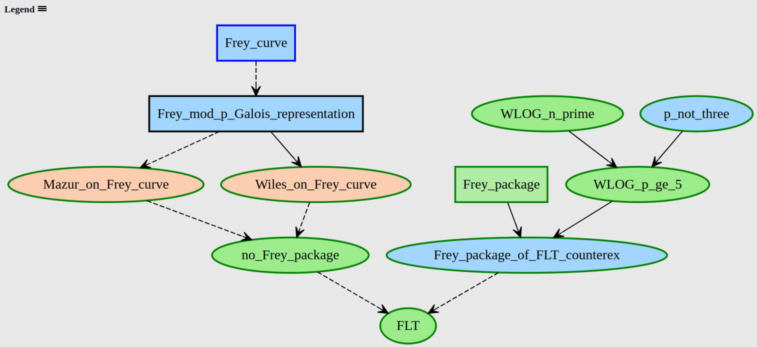


FLT blueprint graph



Green means “done”. Blue means “a nice project (maybe a couple of days)”.

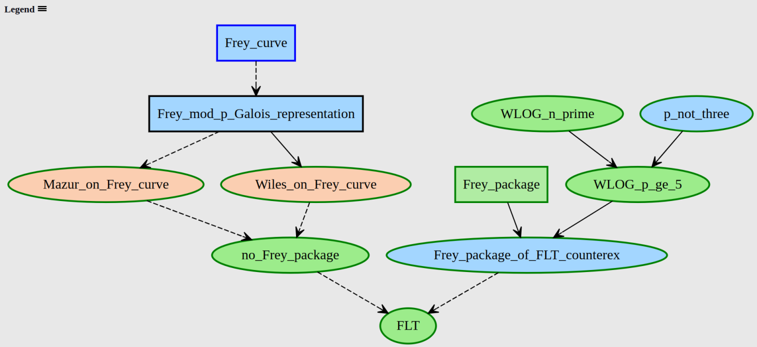
FLT blueprint graph



Green means “done”. Blue means “a nice project (maybe a couple of days)”.

Orange means “a gigantic research project”.

FLT blueprint graph



Green means “done”. Blue means “a nice project (maybe a couple of days)”.

Orange means “a gigantic research project”.

We’re going live in April.

Fermat's Last Theorem

`mathlib` has elliptic curves, modular forms, and quaternion algebras.

Fermat's Last Theorem

`mathlib` has elliptic curves, modular forms, and quaternion algebras.

However, it doesn't have finite flat group schemes, deformation theory of Galois representations, Gorenstein and complete intersection rings, automorphic representations on unit groups of quaternion algebras, Tate modules of elliptic curves, Mazur's theorem on torsion subgroups, local/global class field theory, or any $R = T$ theorems.

Fermat's Last Theorem

`mathlib` has elliptic curves, modular forms, and quaternion algebras.

However, it doesn't have finite flat group schemes, deformation theory of Galois representations, Gorenstein and complete intersection rings, automorphic representations on unit groups of quaternion algebras, Tate modules of elliptic curves, Mazur's theorem on torsion subgroups, local/global class field theory, or any $R = T$ theorems.

The grant starts in October 2024 and finishes in September 2029.

Fermat's Last Theorem

`mathlib` has elliptic curves, modular forms, and quaternion algebras.

However, it doesn't have finite flat group schemes, deformation theory of Galois representations, Gorenstein and complete intersection rings, automorphic representations on unit groups of quaternion algebras, Tate modules of elliptic curves, Mazur's theorem on torsion subgroups, local/global class field theory, or any $R = T$ theorems.

The grant starts in October 2024 and finishes in September 2029.

Can the Lean community prove Fermat's Last Theorem in that time?

Fermat's Last Theorem

`mathlib` has elliptic curves, modular forms, and quaternion algebras.

However, it doesn't have finite flat group schemes, deformation theory of Galois representations, Gorenstein and complete intersection rings, automorphic representations on unit groups of quaternion algebras, Tate modules of elliptic curves, Mazur's theorem on torsion subgroups, local/global class field theory, or any $R = T$ theorems.

The grant starts in October 2024 and finishes in September 2029.

Can the Lean community prove Fermat's Last Theorem in that time?

I don't know!

Fermat’s Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat’s Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Fermat's Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat's Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Here I am confident.

Fermat's Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat's Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Here I am confident.

So this will be my first goal.

Fermat’s Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat’s Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Here I am confident.

So this will be my first goal.

When the blueprint goes online in April, I invite you all to help.

Fermat's Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat's Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Here I am confident.

So this will be my first goal.

When the blueprint goes online in April, I invite you all to help.

You *definitely* don't need to know a proof of Fermat's Last Theorem to be able to do so!

Fermat's Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat's Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Here I am confident.

So this will be my first goal.

When the blueprint goes online in April, I invite you all to help.

You *definitely* don't need to know a proof of Fermat's Last Theorem to be able to do so!

You just have to know some mathematics, and how to write code in Lean.

Fermat's Last Theorem

But here is another question: Can the Lean community, within 5 years, *reduce* the proof of Fermat's Last Theorem (a theorem proved in the 1990s) to results which were known by the 1980s?

Here I am confident.

So this will be my first goal.

When the blueprint goes online in April, I invite you all to help.

You *definitely* don't need to know a proof of Fermat's Last Theorem to be able to do so!

You just have to know some mathematics, and how to write code in Lean.

Thanks a lot for your time!