# Teaching a computer algebraic number theory

David Kurniadi Angdinata

London School of Geometry and Number Theory

Tuesday, 25 March 2025

# Who am I?

I am an aspiring algebraic number theorist in the final year of my PhD.

# Who am I?

I am an aspiring algebraic number theorist in the final year of my PhD.

My thesis will be on twisted L-functions of elliptic curves over global fields, which arise in equivariant Birch and Swinnerton-Dyer conjectures.

# Who am I?

I am an aspiring algebraic number theorist in the final year of my PhD.

My thesis will be on twisted L-functions of elliptic curves over global fields, which arise in equivariant Birch and Swinnerton-Dyer conjectures.

In my first year, I was introduced to an interactive theorem prover called Lean, and tried to formalise the Mordell–Weil theorem as a mini-project.

# Who am I?

I am an aspiring algebraic number theorist in the final year of my PhD.

My thesis will be on twisted L-functions of elliptic curves over global fields, which arise in equivariant Birch and Swinnerton-Dyer conjectures.

In my first year, I was introduced to an interactive theorem prover called Lean, and tried to formalise the Mordell–Weil theorem as a mini-project.

Over the past three years, I saw the potential: prominent mathematicians involved in collaborative projects, massive investments from multinational corporations and philanthropists, headlines of popular science journals.

# Who am I?

I am an aspiring algebraic number theorist in the final year of my PhD.

My thesis will be on twisted L-functions of elliptic curves over global fields, which arise in equivariant Birch and Swinnerton-Dyer conjectures.

In my first year, I was introduced to an interactive theorem prover called Lean, and tried to formalise the Mordell–Weil theorem as a mini-project.

Over the past three years, I saw the potential: prominent mathematicians involved in collaborative projects, massive investments from multinational corporations and philanthropists, headlines of popular science journals.

Today, I will share my thoughts so far:

- why do interactive theorem proving?
- why do interactive theorem proving in Lean?
- why do algebraic number theory in Lean?

# Interactive theorem proving

Wikipedia says that an **interactive theorem prover** is a software tool to assist with the development of formal proofs by human–machine collaboration, which involves some sort of interactive proof editor, or other interface, with which a human can guide the search for proofs, the details of which are stored in, and some steps provided by, a computer.

# Interactive theorem proving

Wikipedia says that an **interactive theorem prover** is a software tool to assist with the development of formal proofs by human–machine collaboration, which involves some sort of interactive proof editor, or other interface, with which a human can guide the search for proofs, the details of which are stored in, and some steps provided by, a computer.

I think that **interactive theorem proving** is the experience of writing mathematics in a rigorous language understood by a computer.

# Interactive theorem proving

Wikipedia says that an **interactive theorem prover** is a software tool to assist with the development of formal proofs by human–machine collaboration, which involves some sort of interactive proof editor, or other interface, with which a human can guide the search for proofs, the details of which are stored in, and some steps provided by, a computer.

I think that **interactive theorem proving** is the experience of writing mathematics in a rigorous language understood by a computer.

This includes axioms, definitions, theorems, proofs, and even techniques.

# Interactive theorem proving

Wikipedia says that an **interactive theorem prover** is a software tool to assist with the development of formal proofs by human–machine collaboration, which involves some sort of interactive proof editor, or other interface, with which a human can guide the search for proofs, the details of which are stored in, and some steps provided by, a computer.

I think that **interactive theorem proving** is the experience of writing mathematics in a rigorous language understood by a computer.

This includes axioms, definitions, theorems, proofs, and even techniques.

Behind the scenes, the compiler does some magic in the *kernel* to assess the validity of the mathematics, akin to the role of journal reviewers.

# Interactive theorem proving

Wikipedia says that an **interactive theorem prover** is a software tool to assist with the development of formal proofs by human–machine collaboration, which involves some sort of interactive proof editor, or other interface, with which a human can guide the search for proofs, the details of which are stored in, and some steps provided by, a computer.

I think that **interactive theorem proving** is the experience of writing mathematics in a rigorous language understood by a computer.

This includes axioms, definitions, theorems, proofs, and even techniques.

Behind the scenes, the compiler does some magic in the *kernel* to assess the validity of the mathematics, akin to the role of journal reviewers.

In this sense, formalisations in interactive theorem provers are *verified*, in contrast to computer algebra systems like Magma and SageMath.

# Basic example

Here is a theorem in Lean that $\mathrm{im}(f) \leq \ker(f)$ whenever $g \circ f = 1$.

```
variable {A B C : Type} [Group A] [Group B] [Group C]

def ker (f : A →* B) : Subgroup A where
  carrier  := {a : A | f a = 1}
  one_mem' := by simp
  inv_mem' := by simp
  mul_mem' := by aesop

def im (f : A →* B) : Subgroup B where
  carrier  := {b : B | ∃ a : A, b = f a}
  one_mem' := ⟨1, by simp⟩
  inv_mem' := fun ⟨a, _⟩ ↦ ⟨a⁻¹, by aesop⟩
  mul_mem' := fun ⟨a, _⟩ ⟨b, _⟩ ↦ ⟨a * b, by aesop⟩

theorem im_le_ker {f : A →* B} {g : B →* C} (h : ∀ a : A, g (f a) = 1) :
    im f ≤ ker g := by    -- Goal is ∀ b : B, (∃ a : A, b = f a) → (g b = 1)
  intro b hb              -- Goal is g b = 1
                          -- New hypotheses (b : B) (hb : ∃ a : A, (b = f a))
  rcases hb with ⟨a, ha⟩  -- Goal is g b = 1
                          -- New hypotheses (b : B) (a : A) (ha : b = f a)
  rewrite [ha]            -- Goal is g (f a) = 1
  exact h a               -- No goals!
```

# Notable examples

Historically, they were used to check proofs that drew scepticism.

# Notable examples

Historically, they were used to check proofs that drew scepticism.

- The proof of the four colour theorem by Appel and Haken (1976) involved analysing 1834 reducible configurations of maps manually. Gonthier et al (2005) formalised it in Coq.

# Notable examples

Historically, they were used to check proofs that drew scepticism.

- The proof of the four colour theorem by Appel and Haken (1976) involved analysing 1834 reducible configurations of maps manually. Gonthier et al (2005) formalised it in Coq.
- The proof of the odd order theorem by Feit and Thompson in 1960 involved complicated arguments in group theory spanning 255 pages. Gonthier et al (2012) formalised it in Coq.

# Notable examples

Historically, they were used to check proofs that drew scepticism.

- The proof of the four colour theorem by Appel and Haken (1976) involved analysing 1834 reducible configurations of maps manually. Gonthier et al (2005) formalised it in Coq.
- The proof of the odd order theorem by Feit and Thompson in 1960 involved complicated arguments in group theory spanning 255 pages. Gonthier et al (2012) formalised it in Coq.
- The proof of the Kepler conjecture by Hales (1998) involved solving 100000 linear programming problems. Hales started the **Flyspeck project** (2003) to formalise it in Coq, and completed it in *11 years*.

# Notable examples

Historically, they were used to check proofs that drew scepticism.

- The proof of the four colour theorem by Appel and Haken (1976) involved analysing 1834 reducible configurations of maps manually. Gonthier et al (2005) formalised it in Coq.
- The proof of the odd order theorem by Feit and Thompson in 1960 involved complicated arguments in group theory spanning 255 pages. Gonthier et al (2012) formalised it in Coq.
- The proof of the Kepler conjecture by Hales (1998) involved solving 100000 linear programming problems. Hales started the **Flyspeck project** (2003) to formalise it in Coq, and completed it in *11 years*.
- The field of condensed mathematics was developed by Clausen and Scholze (2019). Scholze started the **liquid tensor experiment** (2020) to formalise a technical lemma on liquid vector spaces in Lean, and Commelin et al completed it in *20 months*.

# Notable examples

Historically, they were used to check proofs that drew scepticism.

- The proof of the four colour theorem by Appel and Haken (1976) involved analysing 1834 reducible configurations of maps manually. Gonthier et al (2005) formalised it in Coq.
- The proof of the odd order theorem by Feit and Thompson in 1960 involved complicated arguments in group theory spanning 255 pages. Gonthier et al (2012) formalised it in Coq.
- The proof of the Kepler conjecture by Hales (1998) involved solving 100000 linear programming problems. Hales started the **Flyspeck project** (2003) to formalise it in Coq, and completed it in *11 years*.
- The field of condensed mathematics was developed by Clausen and Scholze (2019). Scholze started the **liquid tensor experiment** (2020) to formalise a technical lemma on liquid vector spaces in Lean, and Commelin et al completed it in *20 months*.
- The polynomial Freiman–Ruzsa conjecture was proven by Gowers, Green, Manners, and Tao (Nov 2023). Tao started a formalisation project in Lean 4 days later, and completed it in *3 weeks*.

# Current motivations

Nowadays, they have evolved to serve many other purposes.

# Current motivations

Nowadays, they have evolved to serve many other purposes.

- They allow for large-scale collaborations, akin to Gowers's Polymath Project, but organisation is done via a version control system, and the human moderator is replaced with the language compiler. Example: Tao's **equational relations** project.

# Current motivations

Nowadays, they have evolved to serve many other purposes.

- They allow for large-scale collaborations, akin to Gowers's Polymath Project, but organisation is done via a version control system, and the human moderator is replaced with the language compiler. Example: Tao's **equational relations** project.
- They train artificial intelligence by verifying arguments generated by neural networks when presented with mathematical problems. Example: Google Deepmind's **AlphaProof** model.

# Current motivations

Nowadays, they have evolved to serve many other purposes.

- They allow for large-scale collaborations, akin to Gowers's Polymath Project, but organisation is done via a version control system, and the human moderator is replaced with the language compiler. Example: Tao's **equational relations** project.
- They train artificial intelligence by verifying arguments generated by neural networks when presented with mathematical problems. Example: Google Deepmind's **AlphaProof** model.
- They build self-consistent databases of mathematics for search engines, akin to de Jong's Stacks Project in algebraic geometry. Example: Peking University BICMR's **LeanSearch** engine.

# Current motivations

Nowadays, they have evolved to serve many other purposes.

- They allow for large-scale collaborations, akin to Gowers's Polymath Project, but organisation is done via a version control system, and the human moderator is replaced with the language compiler.
  Example: Tao's **equational relations** project.
- They train artificial intelligence by verifying arguments generated by neural networks when presented with mathematical problems.
  Example: Google Deepmind's **AlphaProof** model.
- They build self-consistent databases of mathematics for search engines, akin to de Jong's Stacks Project in algebraic geometry.
  Example: Peking University BICMR's **LeanSearch** engine.
- They present surprising artifacts, such as unnecessary assumptions, simplification of arguments, or even issues in existing literature.
  Example: Chambert-Loir and Frutos-Fernández discovered an incorrect lemma in a fundamental paper on divided power structures (Dec 2024), which temporarily *broke crystalline cohomology*!

# Current motivations

Nowadays, they have evolved to serve many other purposes.

- They allow for large-scale collaborations, akin to Gowers's Polymath Project, but organisation is done via a version control system, and the human moderator is replaced with the language compiler. Example: Tao's **equational relations** project.
- They train artificial intelligence by verifying arguments generated by neural networks when presented with mathematical problems. Example: Google Deepmind's **AlphaProof** model.
- They build self-consistent databases of mathematics for search engines, akin to de Jong's Stacks Project in algebraic geometry. Example: Peking University BICMR's **LeanSearch** engine.
- They present surprising artifacts, such as unnecessary assumptions, simplification of arguments, or even issues in existing literature. Example: Chambert-Loir and Frutos-Fernández discovered an incorrect lemma in a fundamental paper on divided power structures (Dec 2024), which temporarily *broke crystalline cohomology*!
- They are *fun*! Example: I am addicted.

# Lean theorem prover

Lean is one of the many interactive theorem provers commonly used for formalising pure mathematics: Isabelle, HOL Light, Coq, Metamath, Mizar, etc.

# Lean theorem prover

Lean is one of the many interactive theorem provers commonly used for formalising pure mathematics: Isabelle, HOL Light, Coq, Metamath, Mizar, etc.

The first version of Lean was launched in 2013 by Leonardo de Moura at *Microsoft Research* and later at *Amazon Web Services*, with current development supported by the *Lean Focused Research Organisation*.

# Lean theorem prover

Lean is one of the many interactive theorem provers commonly used for formalising pure mathematics: Isabelle, HOL Light, Coq, Metamath, Mizar, etc.

The first version of Lean was launched in 2013 by Leonardo de Moura at *Microsoft Research* and later at *Amazon Web Services*, with current development supported by the *Lean Focused Research Organisation*.

The current version of Lean uses a *dependent type theory* called *calculus of constructions with inductive types*, unlike Isabelle and HOL Light.

# Lean theorem prover

Lean is one of the many interactive theorem provers commonly used for formalising pure mathematics: Isabelle, HOL Light, Coq, Metamath, Mizar, etc.

The first version of Lean was launched in 2013 by Leonardo de Moura at *Microsoft Research* and later at *Amazon Web Services*, with current development supported by the *Lean Focused Research Organisation*.

The current version of Lean uses a *dependent type theory* called *calculus of constructions with inductive types*, unlike Isabelle and HOL Light.

Furthermore, it is also a *functional programming language*, written in C++ and *Lean*, with support for *multithreading* and *metaprogramming*.

# Lean theorem prover

Lean is one of the many interactive theorem provers commonly used for formalising pure mathematics: Isabelle, HOL Light, Coq, Metamath, Mizar, etc.

The first version of Lean was launched in 2013 by Leonardo de Moura at *Microsoft Research* and later at *Amazon Web Services*, with current development supported by the *Lean Focused Research Organisation*.

The current version of Lean uses a *dependent type theory* called *calculus of constructions with inductive types*, unlike Isabelle and HOL Light.

Furthermore, it is also a *functional programming language*, written in C++ and *Lean*, with support for *multithreading* and *metaprogramming*.

Finally, it supports *Unicode symbols*, and can be run in *common editors* like Visual Studio Code, Emacs, and Neovim.

# Lean's mathematical community

I think the main appeal of Lean is its mathematical community, which arose from *purely sociological* factors: cleanliness of language syntax, good communication with developers, amazing support from community, launch of collaborative projects, interest from Fields Medallists, etc.

# Lean's mathematical community

I think the main appeal of Lean is its mathematical community, which arose from *purely sociological* factors: cleanliness of language syntax, good communication with developers, amazing support from community, launch of collaborative projects, interest from Fields Medallists, etc.

As a result, the community embarked on many collaborative projects:

- perfectoid spaces by Buzzard, Commelin, and Massot (2018 – 2020)
- sphere eversion by Massot, Nash, and van Doorn (2020 – 2023)
- Fermat's last theorem for regular primes by Best, Birkbeck, Brasca, Rodriguez, van de Velde, and Yang (2023 – 2024)

# Lean's mathematical community

I think the main appeal of Lean is its mathematical community, which arose from *purely sociological* factors: cleanliness of language syntax, good communication with developers, amazing support from community, launch of collaborative projects, interest from Fields Medallists, etc.

As a result, the community embarked on many collaborative projects:

- perfectoid spaces by Buzzard, Commelin, and Massot (2018 – 2020)
- sphere eversion by Massot, Nash, and van Doorn (2020 – 2023)
- Fermat's last theorem for regular primes by Best, Birkbeck, Brasca, Rodriguez, van de Velde, and Yang (2023 – 2024)
- local class field theory by Frutos-Fernández and Nuccio (Sep 2022 –)
- prime number theorem and... led by Kontorovich (Jan 2024 –)
- analytic number theory exponent database led by Tao (Aug 2024 –)
- infinity cosmos led by Riehl (Sep 2024 –)
- Fermat's last theorem led by Buzzard (Oct 2024 –)

# Lean's mathematical library

Large formalisation projects are enabled by Lean's mathematical library `mathlib`, which is completely *monolithic* and emphasises *generality*.

# Lean's mathematical library

Large formalisation projects are enabled by Lean's mathematical library `mathlib`, which is completely *monolithic* and emphasises *generality*.

Currently, `mathlib` is a graduate student in algebra, and knows:

- linear algebra: matrices, projective modules, Clifford algebras, Hopf algebras, Lie algebras, ordinary representation theory

# Lean's mathematical library

Large formalisation projects are enabled by Lean's mathematical library `mathlib`, which is completely *monolithic* and emphasises *generality*.

Currently, `mathlib` is a graduate student in algebra, and knows:

- linear algebra: matrices, projective modules, Clifford algebras, Hopf algebras, Lie algebras, ordinary representation theory
- field theory: finite fields, function fields, finite Galois theory, absolute Galois groups, central division algebras, Ax–Grothendieck theorem

# Lean's mathematical library

Large formalisation projects are enabled by Lean's mathematical library `mathlib`, which is completely *monolithic* and emphasises *generality*.

Currently, `mathlib` is a graduate student in algebra, and knows:

- linear algebra: matrices, projective modules, Clifford algebras, Hopf algebras, Lie algebras, ordinary representation theory
- field theory: finite fields, function fields, finite Galois theory, absolute Galois groups, central division algebras, Ax–Grothendieck theorem
- group theory: symmetric groups, structure theorems, Sylow theorems, nilpotent groups, primitive groups, finitely generated groups, free groups, Coxeter groups, abelian group cohomology

# Lean's mathematical library

Large formalisation projects are enabled by Lean's mathematical library `mathlib`, which is completely *monolithic* and emphasises *generality*.

Currently, `mathlib` is a graduate student in algebra, and knows:

- linear algebra: matrices, projective modules, Clifford algebras, Hopf algebras, Lie algebras, ordinary representation theory
- field theory: finite fields, function fields, finite Galois theory, absolute Galois groups, central division algebras, Ax–Grothendieck theorem
- group theory: symmetric groups, structure theorems, Sylow theorems, nilpotent groups, primitive groups, finitely generated groups, free groups, Coxeter groups, abelian group cohomology
- ring theory: domains, localisation, primary decomposition, integral closure, chain conditions, graded modules, Henselian rings, modules of differentials, basic dimension theory, basic homological algebra

# Lean's mathematical library

Large formalisation projects are enabled by Lean's mathematical library `mathlib`, which is completely *monolithic* and emphasises *generality*.

Currently, `mathlib` is a graduate student in algebra, and knows:

- linear algebra: matrices, projective modules, Clifford algebras, Hopf algebras, Lie algebras, ordinary representation theory
- field theory: finite fields, function fields, finite Galois theory, absolute Galois groups, central division algebras, Ax–Grothendieck theorem
- group theory: symmetric groups, structure theorems, Sylow theorems, nilpotent groups, primitive groups, finitely generated groups, free groups, Coxeter groups, abelian group cohomology
- ring theory: domains, localisation, primary decomposition, integral closure, chain conditions, graded modules, Henselian rings, modules of differentials, basic dimension theory, basic homological algebra
- algebraic geometry: schemes, morphism properties, valuative criteria, coherent sheaves, sheaf cohomology, Grothendieck topologies

# Algebraic number theory in Lean

In particular, `mathlib` also knows some non-trivial results in number theory, which is fundamentally *applied* pure mathematics, including:

- elementary theory: Lagrange's theorem on sums of four squares, Legendre's theorem on rational approximation, Gauss's law of quadratic reciprocity, Bertrand's postulate

# Algebraic number theory in Lean

In particular, `mathlib` also knows some non-trivial results in number theory, which is fundamentally *applied* pure mathematics, including:

- elementary theory: Lagrange's theorem on sums of four squares, Legendre's theorem on rational approximation, Gauss's law of quadratic reciprocity, Bertrand's postulate
- analytic theory: Dirichlet's theorem on primes in arithmetic progressions, Liouville's theorem on transcendental numbers, boundedness of Eisenstein series, functional equation of Hurwitz $\zeta$-functions, Gallagher's ergodic theorem, the Selberg sieve

# Algebraic number theory in Lean

In particular, `mathlib` also knows some non-trivial results in number theory, which is fundamentally *applied* pure mathematics, including:

- elementary theory: Lagrange's theorem on sums of four squares, Legendre's theorem on rational approximation, Gauss's law of quadratic reciprocity, Bertrand's postulate
- analytic theory: Dirichlet's theorem on primes in arithmetic progressions, Liouville's theorem on transcendental numbers, boundedness of Eisenstein series, functional equation of Hurwitz $\zeta$-functions, Gallagher's ergodic theorem, the Selberg sieve
- local theory: Hensel's lemma for $\mathbb{Q}_p$, ramification–inertia formula, basic properties of adèle rings, basic properties of Witt vectors

# Algebraic number theory in Lean

In particular, `mathlib` also knows some non-trivial results in number theory, which is fundamentally *applied* pure mathematics, including:

- elementary theory: Lagrange's theorem on sums of four squares, Legendre's theorem on rational approximation, Gauss's law of quadratic reciprocity, Bertrand's postulate
- analytic theory: Dirichlet's theorem on primes in arithmetic progressions, Liouville's theorem on transcendental numbers, boundedness of Eisenstein series, functional equation of Hurwitz $\zeta$-functions, Gallagher's ergodic theorem, the Selberg sieve
- local theory: Hensel's lemma for $\mathbb{Q}_p$, ramification–inertia formula, basic properties of adèle rings, basic properties of Witt vectors
- global theory: Ostrowski's theorem for $\mathbb{Q}$, product formula, finiteness of class numbers, Dirichlet's unit theorem, Galois group of cyclotomic extensions, Kummer–Dedekind theorem

# Algebraic number theory in Lean

In particular, `mathlib` also knows some non-trivial results in number theory, which is fundamentally *applied* pure mathematics, including:

- elementary theory: Lagrange's theorem on sums of four squares, Legendre's theorem on rational approximation, Gauss's law of quadratic reciprocity, Bertrand's postulate
- analytic theory: Dirichlet's theorem on primes in arithmetic progressions, Liouville's theorem on transcendental numbers, boundedness of Eisenstein series, functional equation of Hurwitz $\zeta$-functions, Gallagher's ergodic theorem, the Selberg sieve
- local theory: Hensel's lemma for $\mathbb{Q}_p$, ramification–inertia formula, basic properties of adèle rings, basic properties of Witt vectors
- global theory: Ostrowski's theorem for $\mathbb{Q}$, product formula, finiteness of class numbers, Dirichlet's unit theorem, Galois group of cyclotomic extensions, Kummer–Dedekind theorem
- Diophantine functions and Matiyasevic's theorem
- Fermat's last theorem for $n = 3, 4$ and for polynomials

# Algebraic number theory in Lean

In particular, `mathlib` also knows some non-trivial results in number theory, which is fundamentally *applied* pure mathematics, including:

- elementary theory: Lagrange's theorem on sums of four squares, Legendre's theorem on rational approximation, Gauss's law of quadratic reciprocity, Bertrand's postulate
- analytic theory: Dirichlet's theorem on primes in arithmetic progressions, Liouville's theorem on transcendental numbers, boundedness of Eisenstein series, functional equation of Hurwitz $\zeta$-functions, Gallagher's ergodic theorem, the Selberg sieve
- local theory: Hensel's lemma for $\mathbb{Q}_p$, ramification–inertia formula, basic properties of adèle rings, basic properties of Witt vectors
- global theory: Ostrowski's theorem for $\mathbb{Q}$, product formula, finiteness of class numbers, Dirichlet's unit theorem, Galois group of cyclotomic extensions, Kummer–Dedekind theorem
- Diophantine functions and Matiyasevic's theorem
- Fermat's last theorem for $n = 3, 4$ and for polynomials

It will be learning about class field theory in July!

# Elliptic curves in Lean

Since 2021, I have been formalising an *algebraic* theory of elliptic curves.

# Elliptic curves in Lean

Since 2021, I have been formalising an *algebraic* theory of elliptic curves.

However, `mathlib` does not know what a curve is! This is actually fine, because the Riemann–Roch theorem gives an equivalence of categories

$$\{\text{elliptic curves over } F\} \cong \left\{ \begin{array}{c} \text{tuples } (a_1, a_2, a_3, a_4, a_6) \in F^5 \\ \text{such that } \Delta(a_i) \neq 0 \end{array} \right\},$$

for any field $F$.

# Elliptic curves in Lean

Since 2021, I have been formalising an *algebraic* theory of elliptic curves.

However, `mathlib` does not know what a curve is! This is actually fine, because the Riemann–Roch theorem gives an equivalence of categories

$$\{\text{elliptic curves over } F\} \cong \left\{ \begin{array}{c} \text{tuples } (a_1, a_2, a_3, a_4, a_6) \in F^5 \\ \text{such that } \Delta(a_i) \neq 0 \end{array} \right\},$$

for any field $F$. Currently `mathlib` thinks that

- an elliptic curve $E$ over a ring $R$ is the data of a tuple $(a_1, a_2, a_3, a_4, a_6) \in R^5$ and a proof that $\Delta(a_i) \in R^\times$, and
- a point on $E$ is either $\mathcal{O}$ or a tuple $(x, y) \in R^2$ such that

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x_2 x^2 + a_4 x + a_6.$$

# Elliptic curves in Lean

Since 2021, I have been formalising an *algebraic* theory of elliptic curves.

However, `mathlib` does not know what a curve is! This is actually fine, because the Riemann–Roch theorem gives an equivalence of categories

$$\{\text{elliptic curves over } F\} \cong \left\{ \begin{array}{c} \text{tuples } (a_1, a_2, a_3, a_4, a_6) \in F^5 \\ \text{such that } \Delta(a_i) \neq 0 \end{array} \right\},$$

for any field $F$. Currently `mathlib` thinks that

- an elliptic curve $E$ over a ring $R$ is the data of a tuple $(a_1, a_2, a_3, a_4, a_6) \in R^5$ and a proof that $\Delta(a_i) \in R^\times$, and
- a point on $E$ is either $\mathcal{O}$ or a tuple $(x, y) \in R^2$ such that

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x_2x^2 + a_4x + a_6.$$

The algebra can be developed independently of the geometry!

How far can we develop the arithmetic purely algebraically?

# The group law in Lean

By construction, this means that the geometric addition law is given by explicit rational functions, and associativity is known to be *very difficult*.

# The group law in Lean

By construction, this means that the geometric addition law is given by explicit rational functions, and associativity is known to be *very difficult*. Generically, an equality of the $X$-coordinates of $(P + Q) + R$ and $P + (Q + R)$ is an equality of multivariate polynomials with 26,082 terms!

# The group law in Lean

By construction, this means that the geometric addition law is given by explicit rational functions, and associativity is known to be *very difficult*. Generically, an equality of the $X$-coordinates of $(P + Q) + R$ and $P + (Q + R)$ is an equality of multivariate polynomials with 26,082 terms!

Classically, Riemann–Roch gives an explicit bijection from $E(F)$ to the degree-zero divisor class group $\mathrm{Pic}_F^0(E)$ that preserves the addition law.

# The group law in Lean

By construction, this means that the geometric addition law is given by explicit rational functions, and associativity is known to be *very difficult*. Generically, an equality of the $X$-coordinates of $(P+Q)+R$ and $P+(Q+R)$ is an equality of multivariate polynomials with 26,082 terms!

Classically, Riemann–Roch gives an explicit bijection from $E(F)$ to the degree-zero divisor class group $\mathrm{Pic}_F^0(E)$ that preserves the addition law. While `mathlib` does not know about divisors, it does know that integral domains $D$ have ideal class groups $\mathrm{Cl}(D)$, which translates to a map

$$\begin{array}{rcl} E(F) & \longrightarrow & \mathrm{Cl}(F[E]) \\ \mathcal{O} & \longmapsto & [\langle 1\rangle] \\ (x,y) & \longmapsto & [\langle X-x, Y-y\rangle] \end{array}.$$

# The group law in Lean

By construction, this means that the geometric addition law is given by explicit rational functions, and associativity is known to be *very difficult*. Generically, an equality of the $X$-coordinates of $(P+Q)+R$ and $P+(Q+R)$ is an equality of multivariate polynomials with 26,082 terms!

Classically, Riemann–Roch gives an explicit bijection from $E(F)$ to the degree-zero divisor class group $\mathrm{Pic}^0_F(E)$ that preserves the addition law. While `mathlib` does not know about divisors, it does know that integral domains $D$ have ideal class groups $\mathrm{Cl}(D)$, which translates to a map

$$\begin{array}{rcl} E(F) & \longrightarrow & \mathrm{Cl}(F[E]) \\ \mathcal{O} & \longmapsto & [\langle 1\rangle] \\ (x,y) & \longmapsto & [\langle X-x, Y-y\rangle] \end{array}.$$

In 2022, Junyan Xu discovered an elementary but novel proof that this map is injective, *due to limitations in* `mathlib`. I formalised his argument in Lean and we wrote a paper that was published in ITP 2023.

# The $n$-torsion subgroup in Lean

The fundamental theorem in the *arithmetic* of elliptic curves is the fact that $E_{\overline{F}}[n]$ is a rank two module over $(\mathbb{Z}/n)[G_F]$ whenever $\mathrm{char}(F) \nmid n$.

# The $n$-torsion subgroup in Lean

The fundamental theorem in the *arithmetic* of elliptic curves is the fact that $E_{\overline{F}}[n]$ is a rank two module over $(\mathbb{Z}/n)[G_F]$ whenever $\mathrm{char}(F) \nmid n$.

Thus the $\ell$-adic Tate module $T_\ell E_{\overline{F}} := \varprojlim_n E_{\overline{F}}[\ell^n]$ is a two-dimensional $\ell$-adic Galois representation, which is crucial in Tate's isogeny theorem, Serre's open image theorem, Wiles's modularity lifting theorem, etc.

# The $n$-torsion subgroup in Lean

The fundamental theorem in the *arithmetic* of elliptic curves is the fact that $E_{\overline{F}}[n]$ is a rank two module over $(\mathbb{Z}/n)[G_F]$ whenever $\mathrm{char}(F) \nmid n$.

Thus the $\ell$-adic Tate module $T_\ell E_{\overline{F}} := \varprojlim_n E_{\overline{F}}[\ell^n]$ is a two-dimensional $\ell$-adic Galois representation, which is crucial in Tate's isogeny theorem, Serre's open image theorem, Wiles's modularity lifting theorem, etc.

The Arithmetic of Elliptic Curves by Silverman attempts to prove this in Exercise 3.7 in seven parts, providing explicit inductive definitions of certain division polynomials $\psi_n, \phi_n, \omega_n \in F[X, Y]$ in terms of $a_i \in F$.

# The $n$-torsion subgroup in Lean

The fundamental theorem in the *arithmetic* of elliptic curves is the fact that $E_{\overline{F}}[n]$ is a rank two module over $(\mathbb{Z}/n)[G_F]$ whenever $\mathrm{char}(F) \nmid n$.

Thus the $\ell$-adic Tate module $T_\ell E_{\overline{F}} := \varprojlim_n E_{\overline{F}}[\ell^n]$ is a two-dimensional $\ell$-adic Galois representation, which is crucial in Tate's isogeny theorem, Serre's open image theorem, Wiles's modularity lifting theorem, etc.

The Arithmetic of Elliptic Curves by Silverman attempts to prove this in Exercise 3.7 in seven parts, providing explicit inductive definitions of certain division polynomials $\psi_n, \phi_n, \omega_n \in F[X, Y]$ in terms of $a_i \in F$.

Exercise 3.7(d) claims that for any point $(x, y) \in E(F)$,

$$[n]((x, y)) = \left(\frac{\phi_n(x, y)}{\psi_n(x, y)^2}, \frac{\omega_n(x, y)}{\psi_n(x, y)^3}\right).$$

The key idea is that $\psi_n(x, y) = 0$ occurs precisely when $[n]((x, y)) = \mathcal{O}$.

# The multiplication-by-$n$ formula in Lean

**Conjecture**

*No one has done Exercise 3.7 purely algebraically.*

# The multiplication-by-$n$ formula in Lean

## Conjecture

*No one has done Exercise 3.7 purely algebraically.*

## Proof.

- Exercise 3.7(c) that $(\phi_n, \psi_n^2) = 1$ needs Exercise 3.7(d).

# The multiplication-by-$n$ formula in Lean

## Conjecture

*No one has done Exercise 3.7 purely algebraically.*

## Proof.

- Exercise 3.7(c) that $(\phi_n, \psi_n^2) = 1$ needs Exercise 3.7(d).
- Definition of $\omega_n$ is *incorrect*! It should instead be

$$\omega_n := \frac{1}{2}\left(\psi_{2n}/\psi_n - a_1\phi_n\psi_n - a_3\psi_n^3\right).$$

# The multiplication-by-$n$ formula in Lean

## Conjecture

*No one has done Exercise 3.7 purely algebraically.*

## Proof.

- Exercise 3.7(c) that $(\phi_n, \psi_n^2) = 1$ needs Exercise 3.7(d).
- Definition of $\omega_n$ is *incorrect*! It should instead be

$$\omega_n := \frac{1}{2}\left(\psi_{2n}/\psi_n - a_1\phi_n\psi_n - a_3\psi_n^3\right).$$

- Integrality of $\omega_n$ needs Exercise 3.7(g) that $\psi_n$ is an elliptic sequence

$$\psi_{n+m}\psi_{n-m}\psi_r^2 = \psi_{n+r}\psi_{n-r}\psi_m^2 - \psi_{m+r}\psi_{m-r}\psi_n^2.$$

# The multiplication-by-$n$ formula in Lean

## Conjecture

*No one has done Exercise 3.7 purely algebraically.*

## Proof.

- Exercise 3.7(c) that $(\phi_n, \psi_n^2) = 1$ needs Exercise 3.7(d).
- Definition of $\omega_n$ is *incorrect*! It should instead be

$$\omega_n := \frac{1}{2}\left(\psi_{2n}/\psi_n - a_1\phi_n\psi_n - a_3\psi_n^3\right).$$

- Integrality of $\omega_n$ needs Exercise 3.7(g) that $\psi_n$ is an elliptic sequence

$$\psi_{n+m}\psi_{n-m}\psi_r^2 = \psi_{n+r}\psi_{n-r}\psi_m^2 - \psi_{m+r}\psi_{m-r}\psi_n^2.$$

- Exercise 3.7(g) needs the stronger result that $\psi_n$ is an elliptic net

$$\psi_{n+m}\psi_{n-m}\psi_{r+s}\psi_{r-s} = \psi_{n+r}\psi_{n-r}\psi_{m+s}\psi_{m-s} - \psi_{m+r}\psi_{m-r}\psi_{n+s}\psi_{n-s}.$$

# The multiplication-by-$n$ formula in Lean

## Conjecture

*No one has done Exercise 3.7 purely algebraically.*

## Proof.

- Exercise 3.7(c) that $(\phi_n, \psi_n^2) = 1$ needs Exercise 3.7(d).
- Definition of $\omega_n$ is *incorrect*! It should instead be

$$\omega_n := \frac{1}{2}\left(\psi_{2n}/\psi_n - a_1\phi_n\psi_n - a_3\psi_n^3\right).$$

- Integrality of $\omega_n$ needs Exercise 3.7(g) that $\psi_n$ is an elliptic sequence

$$\psi_{n+m}\psi_{n-m}\psi_r^2 = \psi_{n+r}\psi_{n-r}\psi_m^2 - \psi_{m+r}\psi_{m-r}\psi_n^2.$$

- Exercise 3.7(g) needs the stronger result that $\psi_n$ is an elliptic net

$$\psi_{n+m}\psi_{n-m}\psi_{r+s}\psi_{r-s} = \psi_{n+r}\psi_{n-r}\psi_{m+s}\psi_{m-s} - \psi_{m+r}\psi_{m-r}\psi_{n+s}\psi_{n-s}.$$

- Exercise 3.7(d) needs four special cases of this stronger result. $\square$

# The $\ell$-adic Tate module in Lean

Peiran Wu (■), Junyan Xu (■), and I (■) formalised $T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2$ in Lean.

# The $\ell$-adic Tate module in Lean

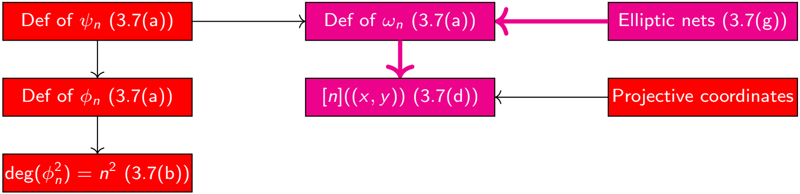
Peiran Wu ($\blacksquare$), Junyan Xu ($\blacksquare$), and I ($\blacksquare$) formalised $T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2$ in Lean.

$$\begin{array}{c}
\text{Def of } \psi_n \text{ (3.7(a))} \\
\downarrow \\
\text{Def of } \phi_n \text{ (3.7(a))} \\
\downarrow \\
\deg(\phi_n^2) = n^2 \text{ (3.7(b))}
\end{array}$$

Projective coordinates

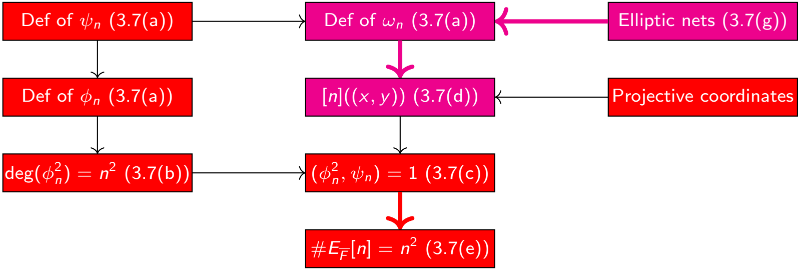
# The $\ell$-adic Tate module in Lean

Peiran Wu ( ), Junyan Xu ( ), and I ( ) formalised $T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2$ in Lean.



# The $\ell$-adic Tate module in Lean

Peiran Wu ( ), Junyan Xu ( ), and I ( ) formalised $T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2$ in Lean.



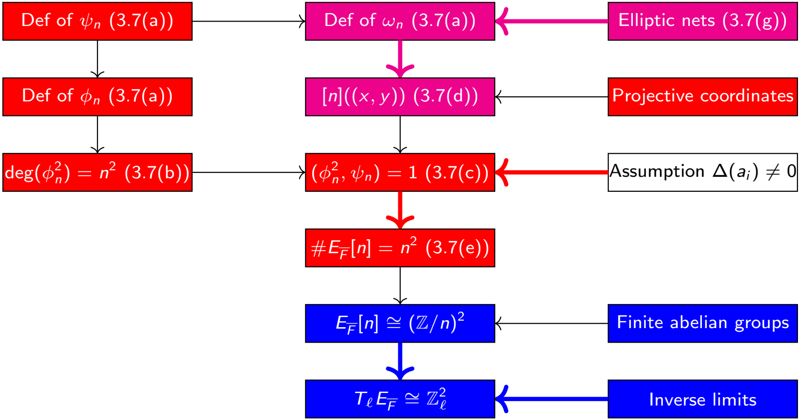
# The $\ell$-adic Tate module in Lean

Peiran Wu (■), Junyan Xu (■), and I (■) formalised $T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2$ in Lean.

$$
\begin{tikzcd}
\text{Def of } \psi_n \text{ (3.7(a))} \arrow[r] \arrow[d] & \text{Def of } \omega_n \text{ (3.7(a))} \arrow[d] & \text{Elliptic nets (3.7(g))} \arrow[l] \\
\text{Def of } \phi_n \text{ (3.7(a))} \arrow[d] & {[n]((x,y))} \text{ (3.7(d))} \arrow[d] & \text{Projective coordinates} \arrow[l] \\
\deg(\phi_n^2) = n^2 \text{ (3.7(b))} \arrow[r] & (\phi_n^2, \psi_n) = 1 \text{ (3.7(c))} \arrow[d] & \\
& \# E_{\overline{F}}[n] = n^2 \text{ (3.7(e))} \arrow[d] & \\
& E_{\overline{F}}[n] \cong (\mathbb{Z}/n)^2 \arrow[d] & \text{Finite abelian groups} \arrow[l] \\
& T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2 & \text{Inverse limits} \arrow[l]
\end{tikzcd}
$$

# The $\ell$-adic Tate module in Lean

Peiran Wu (), Junyan Xu (), and I () formalised $T_\ell E_{\overline{F}} \cong \mathbb{Z}_\ell^2$ in Lean.




Note that the assumption $\Delta(a_i) \neq 0$ is unnecessary until Exercise 3.7(c).

# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?

# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?

$$\begin{tikzcd} E(\mathbb{F}_q) \arrow[dr] & \widehat{E}(\mathcal{M}) \arrow[d] \\ & E(K_{\mathfrak{p}}) \to \widetilde{E}(\kappa_{\mathfrak{p}}) \end{tikzcd}$$

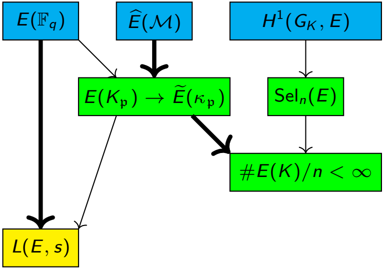
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?

$$\begin{tikzcd}
E(\mathbb{F}_q) \arrow[dd, Rightarrow] \arrow[dr] & \widehat{E}(\mathcal{M}) \arrow[d, Rightarrow] \\
& E(K_{\mathfrak{p}}) \to \widetilde{E}(\kappa_{\mathfrak{p}}) \arrow[dl] \\
L(E, s) &
\end{tikzcd}$$

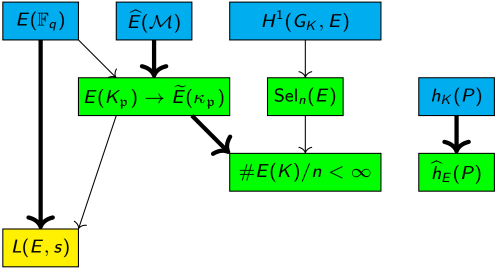
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



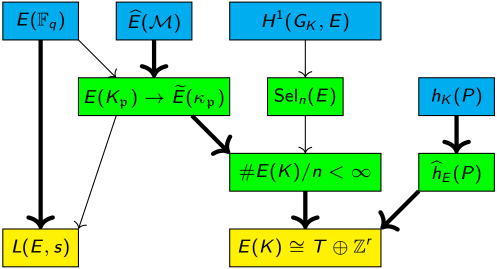
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



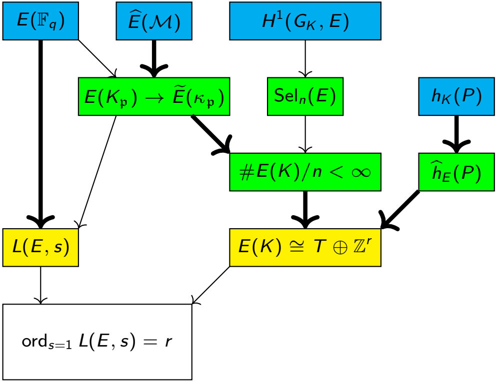
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



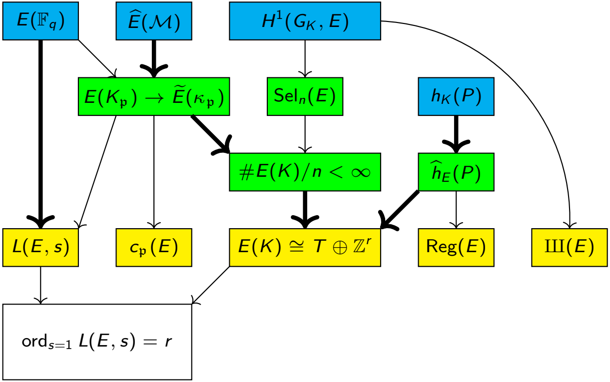
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



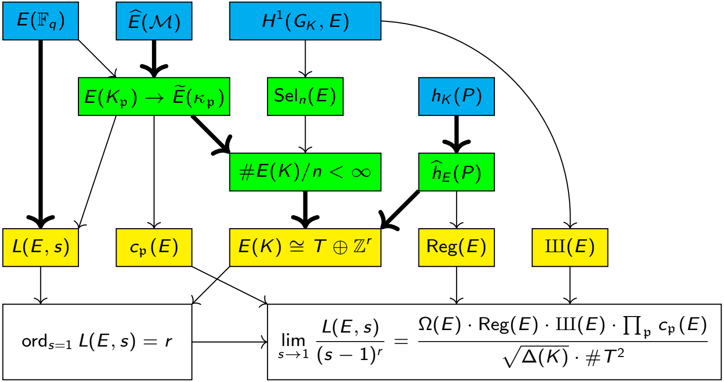
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



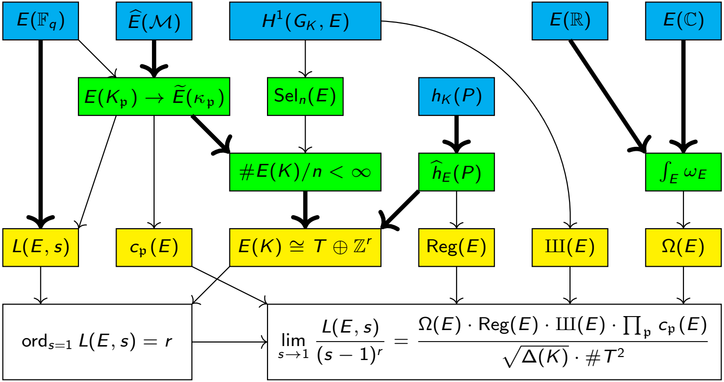
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



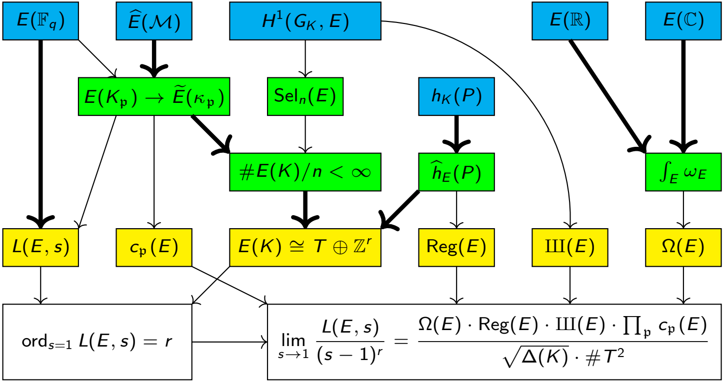
# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?



# Future projects

Can we formalise the *full statement* of the Birch and Swinnerton-Dyer conjecture for an elliptic curve $E$ over a number field $K$ in Lean?




Come join the Lean community to enter a new era of mathematics!