Large-Scale Formal Proof for the Working Mathematician: Lessons Learnt from ALEXANDRIA

Lawrence Paulson, Computer Laboratory, University of Cambridge

CICM, Emmanuel College, Cambridge, 5 September 2023

Background

The formalisation of maths: some history

- *Euclid*: unifying Greek geometry under an axiomatic system
- *Cauchy, Weierstrass*: removing infinitesimals from analysis (and more)
- *Dedekind, Cantor, Frege, Zermelo*: set theory and the axiom of choice
- *Whitehead, Russell, Bourbaki*: formal (or super-rigorous) mathematics
- *de Bruijn*: the AUTOMATH type theory and proof checker; also *Trybulec* and Mizar

Now it's widely accepted that <u>all</u> mathematics is formalisable

But is all maths really formalisable?

As to the question what part of mathematics can be written in AUTOMATH, it should first be remarked that we do not possess a workable definition of the word "mathematics".

Quite often a mathematician jumps from his mathematical language into a kind of metalanguage, obtains results there, and uses these results in his original context. It seems to be very hard to create a single language in which such things can be done without any restriction. — NG de Bruijn, 1968

2017: “Big Proof” (Newton Institute)

- *bringing proof technology into mathematical practice*
- inspired by past formalisations successes: Kepler conjecture, four colour theorem, odd order theorem
- with a focus on *homotopy type theory*
- attendees included Jeremy Avigad, Kevin Buzzard, Tom Hales, Vladimir Voevodsky

Also 2017: ALEXANDRIA

(ERC Project GA 742178)

Aim: to support working mathematicians

… by developing tools and libraries

What areas of mathematics can we formalise?

What sorts of proofs can we formalise?

Project plan

- hire a couple of mathematicians
- formalise a wide variety of mathematical topics
- identify and try to remedy obstacles
- also try AI for search and autoformalisation

All based on Isabelle/HOL

Formalising Mathematics

Mathematics in Isabelle/HOL

- Lots formalised already
- But... was it *sophisticated* enough? *Modern* enough?
- We had to **explore our boundaries**, and compare with *dependent type theories*

Matrix theory, e.g. Perron–Frobenius

Analytic number theory, e.g. Hermite–Lindemann

Homology theory

Measure, integration and probability theory

Complex analysis: residue theorem, prime number theorem

Some warmup formalisations

- *Irrational rapidly convergent series*, formalising a 2002 paper by J. Hančl
- *projective geometry* and *quantum computing*
- counting real and complex roots of polynomials; *Budan-Fourier theorem*

Our focus: recent, sophisticated or potentially problematical material

Another early experiment (2019): algebraically closed fields

Every field admits an algebraically closed extension

(Example: adjoining a root of $x^2 + 1$ to $\mathbb{R}$ to get $\mathbb{C}$)

In general, a *limit* of field extensions

$$K = E_0 \to E_1 \to E_2 \to \cdots \to E_n \to \cdots$$

obtained by adjoining roots. We can form this limit using Zorn's lemma

The work of two summer students, Paulo de Vilhena and Martin Baillon, and the first formalisation of this result in any system.

Taking over a special issue of *Experimental Mathematics*

- *Irrationality and transcendence criteria for infinite series*, incorporating Erdős–Straus and Hančl–Rucki
- *Ordinal partition theory*: delicate constructions by Erdős–Milner and Larson on set-theoretic combinatorics
- *Grothendieck schemes*: answering a challenge by Kevin Buzzard (and completed on the first attempt)

These formed 3 of the 6 papers in the special issue

Upping our ambitions

- extremal graph theory
- additive combinatorics
- combinatorial block designs
- graduate-level number theory
- strict ω-categories

Szemerédi's regularity lemma, and Roth on arithmetic progressions

For every $\epsilon > 0$, there exists a constant M such that every graph has an ϵ-regular partition of its vertex set into at most M parts.

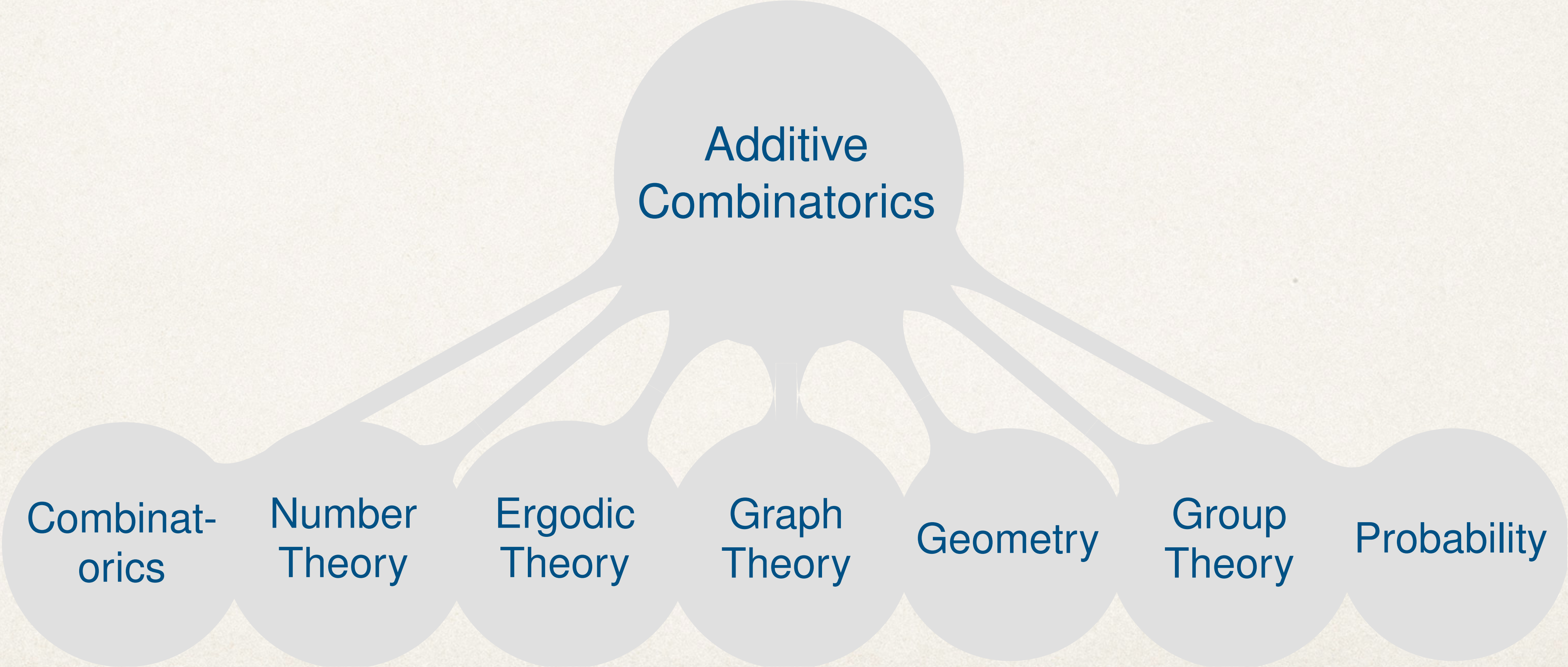
An ϵ-regular partition is where the edges between different parts behave almost randomly when considering subsets of those parts

It is the key tool in the study of large graphs, with applications to algorithm design as well as number theory.

Every subset of the integers with positive *upper asymptotic density* contains a 3-term arithmetic progression.

Additive combinatorics

The study of the additive structure of sets, with numerous applications across mathematics



We study the *sumset* $A + B = \{a + b : a \in A, b \in B\}$

for a given abelian group $(G, +)$

and the *iterated sumset*: the n-fold sum $nA = A + \cdots + A$

Plünnecke–Ruzsa inequality: an upper bound on $mB - nB$

Khovanskii's theorem: $|nA|$ grows like a polynomial for sufficiently large n

Kneser's theorem and the *Cauchy–Davenport theorem: lower bounds for* $|A + B|$

Balog–Szemerédi–Gowers: a deep result bearing on Szemerédi's theorem

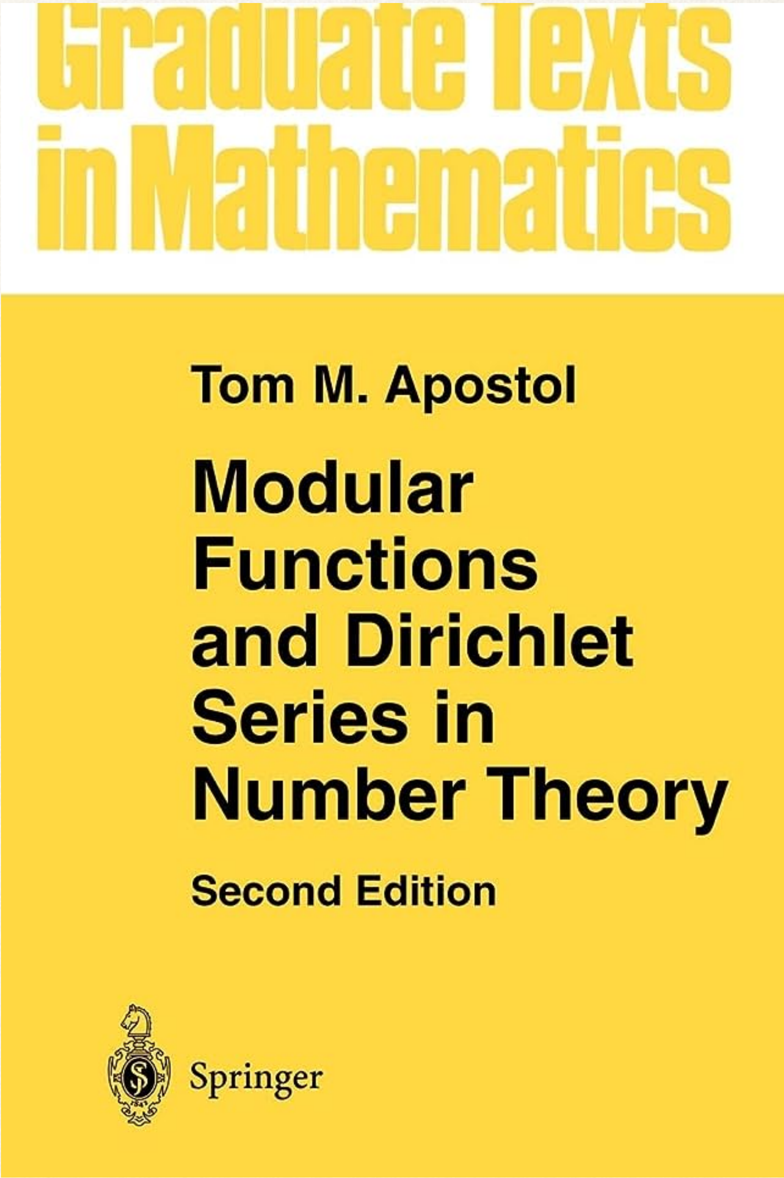
Combinatorial design theory

- dozens of varieties of block designs, hypergraphs, graphs and the relationships among them
- E.g. *Fisher's inequality* for balanced incomplete block designs
- probabilistic and generating function methods
- advanced techniques using Isabelle's *locales*

PhD work of Chelsea Edmonds

Half of a standard number theory text

- Elliptic functions
- The modular group and modular functions
- The Dedekind eta function
- Kronecker's approximation theorem



Lots of advanced material

Definition. For fixed σ, we define

$$m(\sigma) = \inf_t |\zeta(\sigma + it)| \qquad \text{and} \qquad M(\sigma) = \sup_t |\zeta(\sigma + it)|,$$

where the infimum and supremum are taken over all real t.

Theorem 7.11. *For each fixed $\sigma > 1$ we have*

$$M(\sigma) = \zeta(\sigma) \qquad \textit{and} \qquad m(\sigma) = \frac{\zeta(2\sigma)}{\zeta(\sigma)}.$$

Proof. For $\sigma > 1$ we have $|\zeta(\sigma + it)| \le \zeta(\sigma)$ so $M(\sigma) = \zeta(\sigma)$, the supremum being attained on the real axis. To obtain the result for $m(\sigma)$ we estimate the reciprocal $|1/\zeta(s)|$. For $\sigma > 1$ we have

$$\left|\frac{1}{\zeta(s)}\right| = \prod_p |1 - p^{-s}| \le \prod_p (1 + p^{-\sigma}) = \frac{\zeta(\sigma)}{\zeta(2\sigma)}. \tag{17}$$

Hence $|\zeta(s)| \ge \zeta(2\sigma)/\zeta(\sigma)$ so $m(\sigma) \ge \zeta(2\sigma)/\zeta(\sigma)$.

Now we wish to prove the reverse inequality $m(\sigma) \le \zeta(2\sigma)/\zeta(\sigma)$. The idea is to show that the inequality

$$|1 - p^{-s}| \le 1 + p^{-\sigma}$$

used in (17) is very nearly an equality for certain values of t. Now

$$1 - p^{-s} = 1 - p^{-\sigma - it} = 1 - p^{-\sigma}e^{-it\log p} = 1 + p^{-\sigma}e^{i(-t\log p - \pi)},$$

so we need to show that $-t \log p - \pi$ is nearly an even multiple of 2π for certain values of t. For this we invoke Kronecker's theorem. Of course, there are infinitely many terms in the Euler product for $1/\zeta(s)$ and we cannot expect to make $-t \log p - \pi$ nearly an even multiple of 2π for *all* primes p. But we will be able to do this for enough primes to obtain the desired inequality.

On dependently-typed constructions

- Dependent types can be erased from any formal development for working in Isabelle/HOL
- … thereby obtaining legible Isabelle proofs, and benefiting from powerful automation
- Case study: strict ω-categories

[See Bordg & Doña Mateo's paper in CPP 2023]

What does this Work Achieve?

legible, intuitive proofs

no borders between mathematical topics

…and no topics off-limits

performance

Handling sophisticated, modern mathematics

Legible, intuitive proofs

```
lemma sum_diff_split:
  fixes f:: "nat ⇒ 'a::ab_group_add"
  assumes "m ≤ n"
  shows "(∑i≤n - m. f(n - i)) = (∑i≤n. f i) - (∑i<m. f i)"
proof -
  have inj: "inj_on ((-) n) {m..n}"
    by (auto simp: inj_on_def)
  have "(∑i≤n - m. f(n - i)) = (∑i∈(-) n ` {m..n}. f(n - i))"
  proof (rule sum.cong)
    have "⋀x. x ≤ n - m ⟹ ∃k≥m. k ≤ n ∧ x = n - k"
      by (metis assms diff_diff_cancel diff_le_mono2 diff_le_self le_trans)
    then show "{..n - m} = (-) n ` {m..n}"
      by (auto simp: image_iff Bex_def)
  qed auto
  also have "... = (∑i=m..n. f i)"
    by (smt (verit) atLeastAtMost_iff diff_diff_cancel sum.reindex_cong [OF inj])
  also have "... = (∑i≤n. f i) - (∑i<m. f i)"
    using sum_diff_nat_ivl[of 0 "m" "Suc n" f] assms
    by (simp only: atLeast0AtMost atLeast0LessThan atLeastLessThanSuc_atLeastAtMost)
  finally show ?thesis .
qed
```

```
theorem Dirichlet_approx_simult:
  fixes ϑ :: "nat ⇒ real" and N n :: nat
  assumes "N > 0"
  obtains q p where "0<q" "q ≤ int (N^n)" and "⋀i. i<n ⟹ ¦of_int q * ϑ i - of_int(p i)¦ < 1/N"
proof -
  have lessN: "nat ⌊x * real N⌋ < N" if "0 ≤ x" "x < 1" for x
  proof -
    have "⌊x * real N⌋ < N"
      using that by (simp add: assms floor_less_iff)
    with assms show ?thesis by linarith
  qed
  define interv where "interv ≡ λk. {real k/N..< Suc k/N}"
  define fracs where "fracs ≡ λk. map (λi. frac (real k * ϑ i)) [0..<n]"
  define X where "X ≡ fracs ` {..N^n}"
  define Y where "Y ≡ set (List.n_lists n (map interv [0..<N]))"
  have interv_iff: "interv k = interv k' ⟷ k=k'" for k k'
    using assms by (auto simp: interv_def Ico_eq_Ico divide_strict_right_mono)
  have in_interv: "x ∈ interv (nat ⌊x * real N⌋)" if "x≥0" for x
    using that assms by (simp add: interv_def divide_simps) linarith
  have False
    if non: "∀a b. b ≤ N^n ⟶ a < b ⟶ ¬(∀i<n. ¦frac (real b * ϑ i) - frac (real a * ϑ i)¦ < 1/N)"
  proof - [35 lines]
  qed
  then obtain a b where "a<b" "b ≤ N^n" and *: "⋀i. i<n ⟹ ¦frac (real b * ϑ i) - frac (real a * ϑ i)¦ < 1/N"
    by blast
  let ?k = "b-a"
  let ?h = "λi. ⌊b * ϑ i⌋ - ⌊a * ϑ i⌋"
  show ?thesis
  proof
    fix i
    assume "i<n"
    have "frac (b * ϑ i) - frac (a * ϑ i) = ?k * ϑ i - ?h i"
      using ‹a < b› by (simp add: frac_def left_diff_distrib' of_nat_diff)
    then show "¦of_int ?k * ϑ i - ?h i¦ < 1/N"
      by (metis "*" ‹i < n› of_int_of_nat_eq)
  qed (use ‹a < b› ‹b ≤ N^n› in auto)
qed
```

No borders between topics

```
session Modular_Functions (AFP) = Zeta_Function +
  options [timeout = 3600]
  sessions
    "HOL-Library"
    "HOL-Real_Asymp"
    "HOL-Computational_Algebra"
    Formal_Puiseux_Series
    Winding_Number_Eval
    Linear_Recurrences
    Algebraic_Numbers
    Dirichlet_Series
    Dirichlet_L
    Polynomial_Factorization
    Bernoulli
    Landau_Symbols
    Cotangent_PFD_Formula
  theories
    Kronecker_Theorem
    Modular_Functions
    Dedekind_Eta_Function
```

```
theory Khovanskii
  imports
    FiniteProduct
    "Pluennecke_Ruzsa_Inequality.Pluennecke_Ruzsa_Inequality"
    "Bernoulli.Bernoulli"                  — ‹sums of a fixed power are polynomials›
    "HOL-Analysis.Weierstrass_Theorems" — ‹needed for polynomial function›
    "HOL-Library.List_Lenlexorder"        — ‹lexicographic ordering for the type @{typ ‹nat
begin
```

- … and we combined *probability* with *combinatorics*
- … *transfinite recursion* with *holomorphic functions*
- we are perfectly okay without *dependent types*
- with locales we can handle **multiple inheritance** ("diamonds")

Performance matters too!

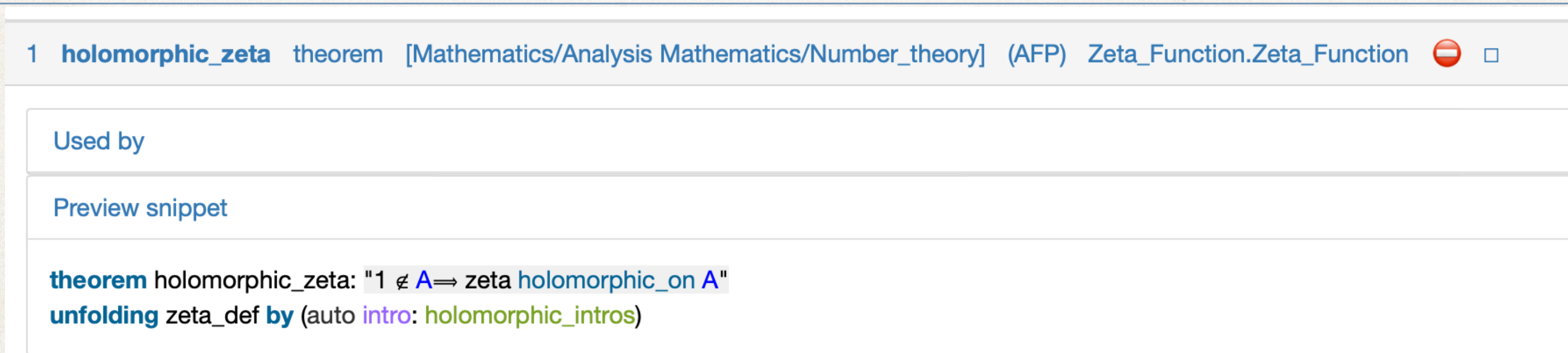
- 0:15 for the Erdős–Straus paper on irrational series
- 1:11 for Balog–Szemerédi–Gowers
- 1:04 for Grothendieck schemes
- 0:50 for ordinal partitions
- 0:14 for Szemerédi's regularity lemma
- 1:03 for Roth's theorem on arithmetic progressions

Run on a 2019 iMac, 3.6 GHz 8-Core Intel Core i9

Search and ML experiments

- The project tasks included
    - Intelligent Search / Proof Idioms
    - Automated User Support
- These were highly speculative ideas about “mining” our existing millions of lines of proofs.

Intelligent Search: SeRAPIS

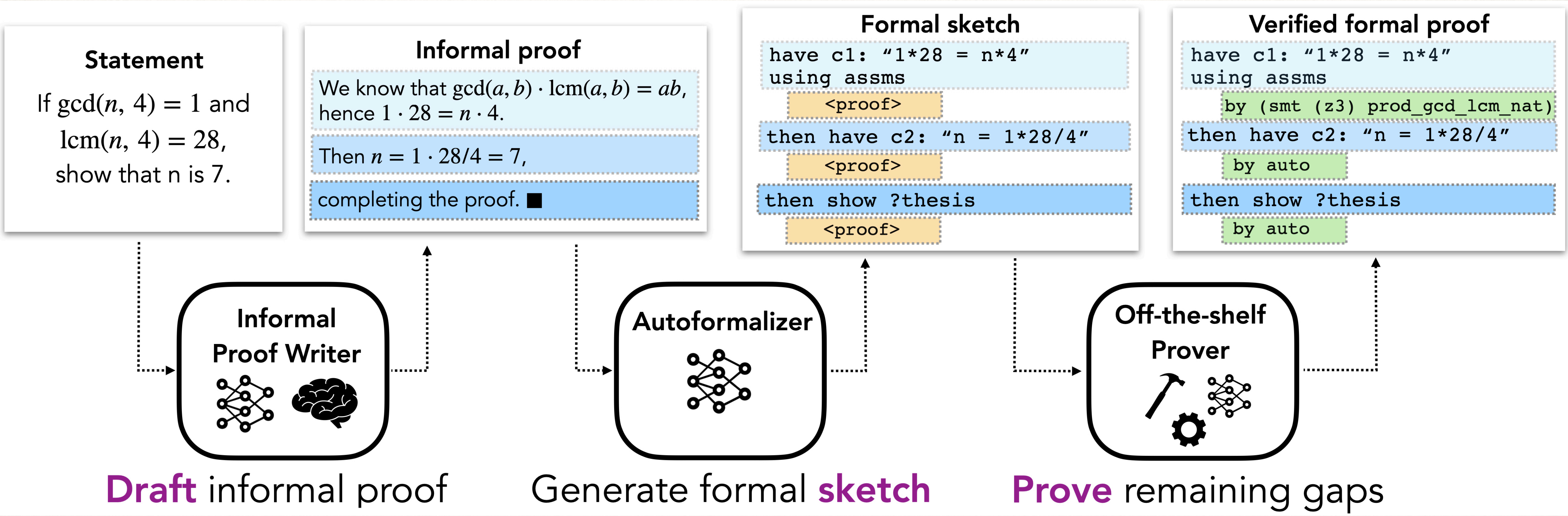


- Quick, concept-oriented search of all Isabelle libraries
- Lots of experimental search options based on a huge index of mathematical terms

ML experiments

- auto-formalisation of text to Isabelle
- *Isabelle Parallel Corpus*, pairing formal theorems and proofs to their natural language counterparts
- generating intermediate goals for proofs
- identifying relevant lemmas

Draft, sketch and prove



Lessons and conclusions

"It is in principle impossible to set up a system of formulas that would be equivalent to intuitionistic mathematics, for the possibilities of thought cannot be reduced to a finite number of rules set up in advance."

– Heyting (1930)

"Thus we are led to conclude that, although everything mathematical is formalisable, it is nevertheless impossible to formalise all of mathematics in a *single* formal system, a fact that intuitionism has asserted all along."

–Kurt Gödel (1935)

- But simple type theory worked fine for practically everything
- *(which means that Whitehead and Russell were right!)*
- We found nothing that we couldn’t formalise (nicely!) — and never had to redo a development
- Although we never had to fight the formalism, newcomers do struggle with *the system*

- We developed *new formalisation methodologies*, especially using **locales**
- We investigated the role of type classes and *type dependency*
- The ML part of the proposal was speculative, but even here the advances are dramatic
- The main obstacles? Gaps in texts, and the sheer immensity of mathematics.

On the other hand…

This never happened!

What areas of mathematics can we formalise?

Everything we tried: combinatorics, number theory, complex analysis, quantum computation, …

What sorts of proofs can we formalise?

Err… Correct proofs that don't have big gaps

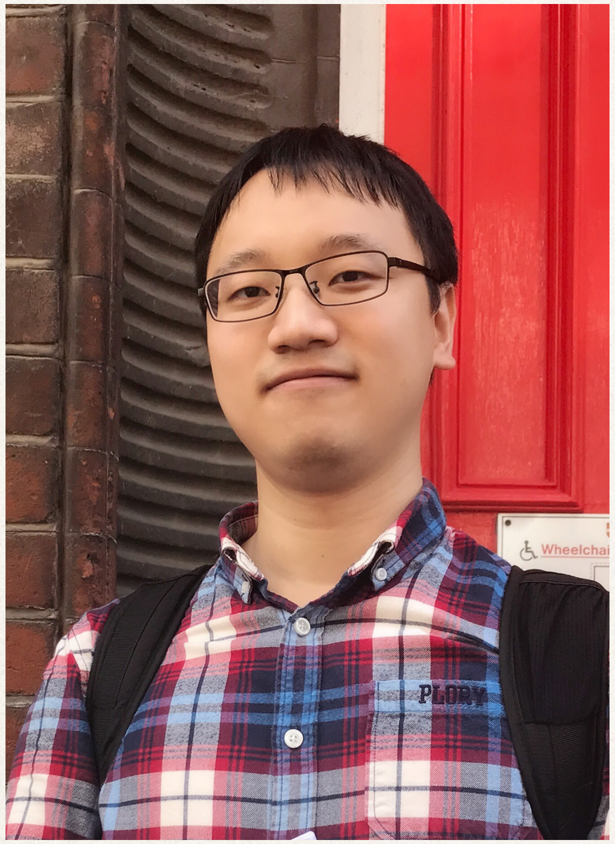
We've formalised the work of two Fields medalists (Roth, Gowers), an Abel prize winner (Szemerédi) … and the legendary Paul Erdős too.

The team

Anthony Bordg
quantum computation, Grothendieck schemes, ω-categories, ML experiments

Angeliki Koutsoukou-Argyraki
Szemerédi & Roth, additive combinatorics, transcendence and irrationality, ML experiments

The team

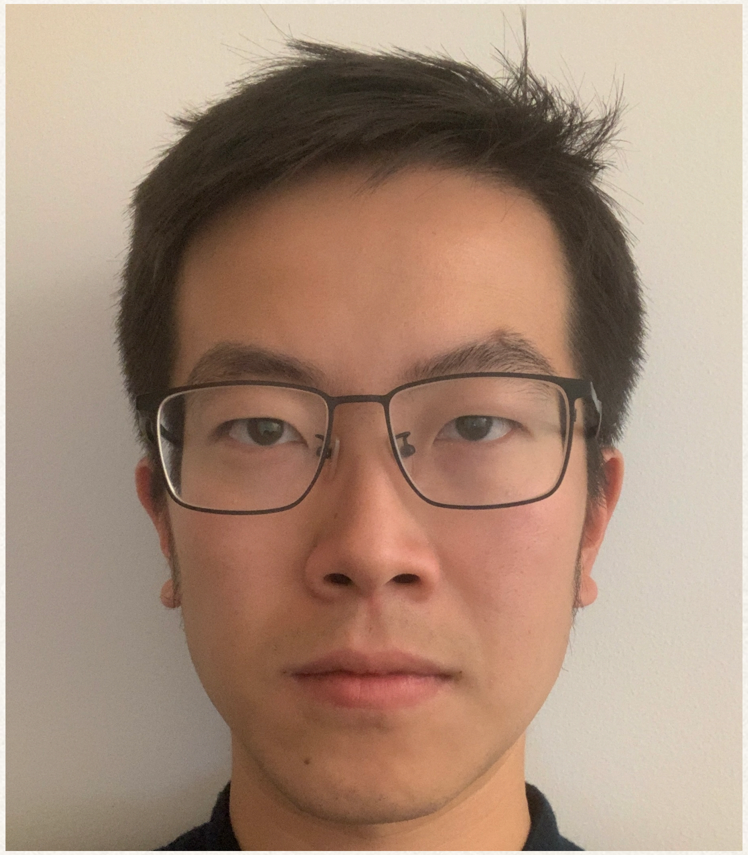


Wenda Li
polynomial roots, ML experiments, transcendence and irrationality, Grothendieck schemes

Yiannos Stathopoulos
SErAPIS search engine, Isabelle parallel corpus, extensive ML experiments

… and PhD students!

Chelsea Edmonds
combinatorial block designs, Balog–Szemerédi–Gowers theorem, Szemerédi & Roth, Lucas's theorem



Albert Qiaochu Jiang
autoformalisation, premise selection, draft/sketch/prove

Other students and interns

Adrián Doña Mateo
Artem Khovanov
Fox Thomson
Hanna Lachnitt
Jamie Chen
Kevin Lee
Mantas Bakšys
Marco Dos Santos
Martin Baillon
Nicolò Cavalleri

Nils Lauermann
Paulo Emílio de Vilhena
Ryan Shao
Xiao Ma
Yaël Dillies
Yijun He
Zhengkun Ye
Zibo Yang