Selling computers to mathematicians.

K. Buzzard, Imperial College London

19th July, Cortona

Before we start

Thank you to the organisers for the invitation, and thanks to you all for showing up.

Stuff in this talk

- Observations on recent events.
- The 2-adic eigencurve at the boundary of weight space.
- How to prove Fermat's Last Theorem in a theorem prover.

Theorem provers at the ICM

There was this International Congress of Mathematicians last week.

I gave a talk about theorem provers to a bunch of people who are interested in the mathematics which is winning Fields Medals.

I am doing OK at raising the profile of computer theorem provers amongst human theorem provers.

The Liquid Tensor Experiment

Also last week, Johan Commelin and his team completed the formal verification of the fundamental theorem underlying the theory of liquid vector spaces, and thus completing Peter Scholze's "Liquid Tensor Experiment" challenge.

Links to the challenge, the announcement of the solution, and an article in Nature about it.

The challenge was to all the prover communities.

Our solution used Lean, a non-univalent prover.

As far as I know, we were the only community who even tried.

What is *important?*

If we want this area to grow, we need to *sell it*.

Explaining to mathematicians why computers are relevant is an important part of the story.

Voevodsky was a brilliant algebraic geometer, and a brilliant type theorist, but I think he was a lousy salesman.

What is *important?*

I think that demonstrating that the systems can engage with the mathematics explained in the lectures at the recent ICM is *important*.

Why? If the people at the top take the area seriously then the area will grow.

We claim that one day computers will help humans to prove theorems in all areas of pure mathematics. Right now computers can't even *understand* most of it. But they can understand the definition of a perfectoid space.

Next step: *stating* important theorems.

My project student Jamie Bell formalised the statement of the BSD conjecture in Lean. Perhaps the first person in the world to do so in any theorem prover?

Marketing HoTT

I find it *really hard* to market the HoTT systems to the people interested in the ICM.

Conversely, I know that for many of you these people are not your target audience.

Someone in your community once told me that there was no point making the stuff `mathlib` and the Liquid Tensor Experiment made because (a) it had been done already and (b) univalence wasn't going to help with those kinds of problems anyway.

I think that "here is a tool which might change mathematics" is a more powerful message than "here are questions about type theory which we can solve on a computer".

Coleman theory

This is a story about a computation. I'm telling it because I can't work out if there's a moral in it somewhere.

In the late 70s/early 80s, Hida invented Hida theory.

Mazur and Wiles computed some explicit examples of Hida families.

In the mid-90s, Coleman invented Coleman theory, a generalisation of Hida theory.

I learnt Coleman theory by reading a huge book on p-adic geometry and then working out an example. The story is about the example.

The eigencurve

The Coleman-Mazur eigencurve $C(N,p)$ is a piece of data (a curve over the p-adic numbers defined by power series equations) depending on a positive natural N and a prime number p.

The example: I once computed the equations on a computer for the boundary of the eigencurve $C(1,2)$ to $O(w^{30},u^{30},2^{30})$ using `pari-gp` and later on `magma` (computer algebra systems).

I noticed a pattern in the answer. "Slopes at points near the boundary of weight space are in arithmetic progression?!"

The eigencurve

I proved some special cases of this (for specific w near the boundary).

I told my PhD student Lloyd Kilford about the problem.

One day he came into my office and remarked that he had computationally noticed a second pattern after computing a bunch of 30×30 matrices to very large 2-adic precision.

I had had no idea that the second pattern was occurring.

2 days later we'd proved everything for $C(1,2)$ and we had a general conjecture for $C(N,p)$.

All coming out of a computation (which proved nothing).

Langlands functoriality

A couple of years ago Newton and Thorne proved symmetric power functoriality for regular algebraic cuspidal automorphic representations of $GL(2)/\mathbb{Q}$.

In their proof they needed a theorem about $C(1,p)$, for any prime p, to prove a statement which didn't mention primes. The Buzzard-Kilford pattern implied the theorem they needed for $C(1,2)$.

A large calculation helped a profound theorem along its way.

Constructivism

Being able to compute was *really important* because it gave us new insights.

But it didn't matter that the computation was unverified. It wasn't going to prove anything anyway. It was only used to *inspire*.

The breakthrough work is the Langlands work of Newton and Thorne.

Computation played no obvious role in their work.

How to prove Fermat's Last Theorem

Fermat's Last Theorem is deduced from the Taniyama–Shimura–Weil conjecture, a special case of the Langlands Philosophy for $GL_2/\mathbb{Q}$.

The proof of the TSW conjecture needs local and global class field theory, which is the Langlands Philosophy for GL_1.

Here's a crash course in local and global class field theory.

Much of it is not formalised in any theorem prover. Anyone interested?

Monoids acting on abelian groups

You start off with a monoid G, with group law $*$.

And you let it act on a set A, so there's axioms like $(g * h) \cdot a = g \cdot (h \cdot a)$ and $1 \cdot a = a$.

And then you let A be an abelian group, with group law $+$.

And you add in the axiom $g \cdot (a + b) = g \cdot a + g \cdot b$.

NB if $G = A$ we're inventing ring theory (exercise: spot the axioms of ring theory hidden above).

Group cohomology.

Set-up: group G acts on abelian group A.

Final ingredient: a natural number n.

Output of group cohomology machine: an additive abelian group $H^n(G, A)$.

We don't have these in Lean's maths library. I don't know of any formalisation of this basic cohomology theory.

Group cohomology.

Variant (Tate): if G is finite then you can input an integer m and get $\widehat{H}^m(G, A)$ which agree for $m \geq 1$ with $H^n(G, A)$.

An API for group cohomology should involve

- Being able to calculate examples like $H^1(\mu_2, \mathbb{Z}/2\mathbb{Z})$;
- Fancy cohomological facts like the Hochschild-Serre spectral sequence.

Continuous group cohomology

Variant: profinite topological groups acting on discrete modules.

- Nice goal: computation of $H^2(\widehat{\mathbb{Z}}, \mathbb{Z})$.

That was algebra. Now some number theory.

Number theory

Now let K be the p-adic numbers $\mathbb{Q}_p$ (if you want to be concrete) or a finite extension of the p-adic numbers (if you want to be abstract).

Say L/K is a finite Galois extension (e.g. choose a random polynomial $f \in K[X]$ and throw in all its roots).

Set $G = Gal(L/K)$, a finite group of size n. Claim: $H^2(G, L^\times)$ is cyclic of order n and has a canonical generator $u_{L/K}$.

Theorem: cupping with $u_{L/K}$ induces an isomorphism $\widehat{H}^m(G, \mathbb{Z}) = \widehat{H}^{m+2}(G, L^\times)$.

Question: Is $u_{L/K}$ "constructive"? Is there a "formula" for it?

Part of the definition looks like this: "general nonsense gives us a constructive map $H_1 \to H_2$ and a proof that it's a bijection; now consider its inverse".

Number theory

Global question.

One can "complete" the rationals with respect to a norm.

One gets the real numbers this way, and also the p-adic numbers.

The same is true for finite extensions of the rationals (e.g. completing $\mathbb{Q}(i)$ gives $\mathbb{C}$).

The completions are topological rings and they have closed unit discs.

Adeles

Let K be $\mathbb{Q}$ or a finite extension.

Consider the product $\prod_v K_v$ of all the completions of K.

Inside this product, consider the subring generated by K and the product of the closed unit discs. Call this subring $\mathbb{A}_K$. It's a K-algebra.

Theorem: $\mathrm{Gal}(\overline{K}/K)^{\mathrm{ab}} \cong \pi_0(K^\times \backslash \mathbb{A}_K^\times)$.

NB: it's not a theorem. It's a definition and a theorem.

Modern proofs use those cohomology elements.

Wiles' proof uses this stuff and nobody is anywhere near formalising it. Why not?

Algebraic geometry

Note: I waffled so much in my talk that I didn't actually get this far; the actual talk stopped on the previous slide. Here are the rest of the notes anyway.

There are things called commutative rings with a 1.

If R is a commutative ring with 1 then Grothendieck wants to define a topological space called $\mathrm{Spec}(R)$, whose underlying type is the prime ideals of R.

$\mathrm{Spec}(R)$ is called an *affine scheme*.

You can glue affine schemes together to make *schemes*.

Algebraic geometry

Making $\mathrm{Spec}(R)$ is problematic constructively.

To prove the intersection of two open sets is open, you need to prove that $IJ \subseteq P$ implies $I \subseteq P$ or $J \subseteq P$.

It's easy to prove $I \not\subseteq P$ and $J \not\subseteq P$ implies $IJ \not\subseteq P$.

Algebraic geometry

Whenever I raise this with constructivists I am assured that there are *other ways to do algebraic geometry*.

That's fine by me! Do it another way. But it would be great if someone other than the Lean community did it.

Here's some goals in algebraic geometry.

Spec is a functor from the category of rings to the category of schemes. "Global sections" is a functor from the category of schemes to the category of rings. Construct these functors and prove they're adjoints.

We have this in Lean via the nonconstructive approach.

Commutative algebra

Given a morphism $f : A \to B$ of rings, you get a morphism of schemes $\mathrm{Spec}(B) \to \mathrm{Spec}(A)$.

There are *loads* of predicates that you can put on morphisms of rings (flat, smooth, unramified, finite type, finitely presented,...).

If these predicates are *local* then they extend to predicates on morphisms of schemes.

Algebraic geometry

This page on the Stacks Project website lists a bunch of properties of morphisms of schemes.

Most of them we don't have in Lean but we're working on them.

I would welcome a competing formalisation.

Conclusion

I've suggested some projects which I could market to mathematicians and which *might* be appropriate to do in a univalent system.

Is anyone interested? Alternatively are there people interested in explaining to me why these are not appropriate problems?

I will continue to try and understand Voevodsky's vision of "doing all of mathematics in a computer proof system" but this is not what I am seeing in practice coming from most of the proof assistant communities.

Thank you for putting up with me.