Computers and mathematics

Damiano Testa

University of Warwick

Warwick Maths Society, December 8th, 2020

Splitting bills

Mathematics is often associated to the idea of performing long and complicated calculations.

For instance, at the end of a dinner, my friends turn to me, asking "How much should we pay?"

I *do* know the answer: "Divide the total by the number of people!"

This has never been the answer they expected.

Eventually, they realize that a phone can answer their question.

Thinking about a new mathematical problem

This situation summarizes well what happens when I am working.

Initial phase: thinking about a new problem, exploring possible solutions...

In the "going out with friends" analogy, this is the part that I enjoy the most: having fun, laughing, sharing a meal...

The bill

After that, comes the bill: I know the problem and the steps involved in its solution.

All that is left is to write it down to find hidden mistakes.

Just like with paying the bill, this is an important part of the process.

Help with the bill

Writing a solution helps to expose further flaws and mistakes.

Hopefully, these can be fixed, the write up goes through and you are now the author of a mathematical paper!

To split the bill, I welcome the help of my friends, especially when they pull out a phone!

What tools are available to help verifying the correctness of our reasoning?

Tools

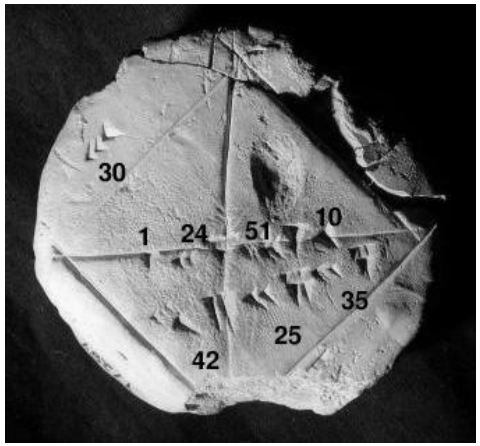
Pen and paper (or clay tablets, wall inscriptions, word processing, LaTeX,...)

Annotate ideas, symbols and partial computations, make drawings...

Writing arguments formally to check implications is a great way of finding mistakes and flaws in our proofs.

Also, performing calculations on paper is much more reliable than simply performing them in our minds.

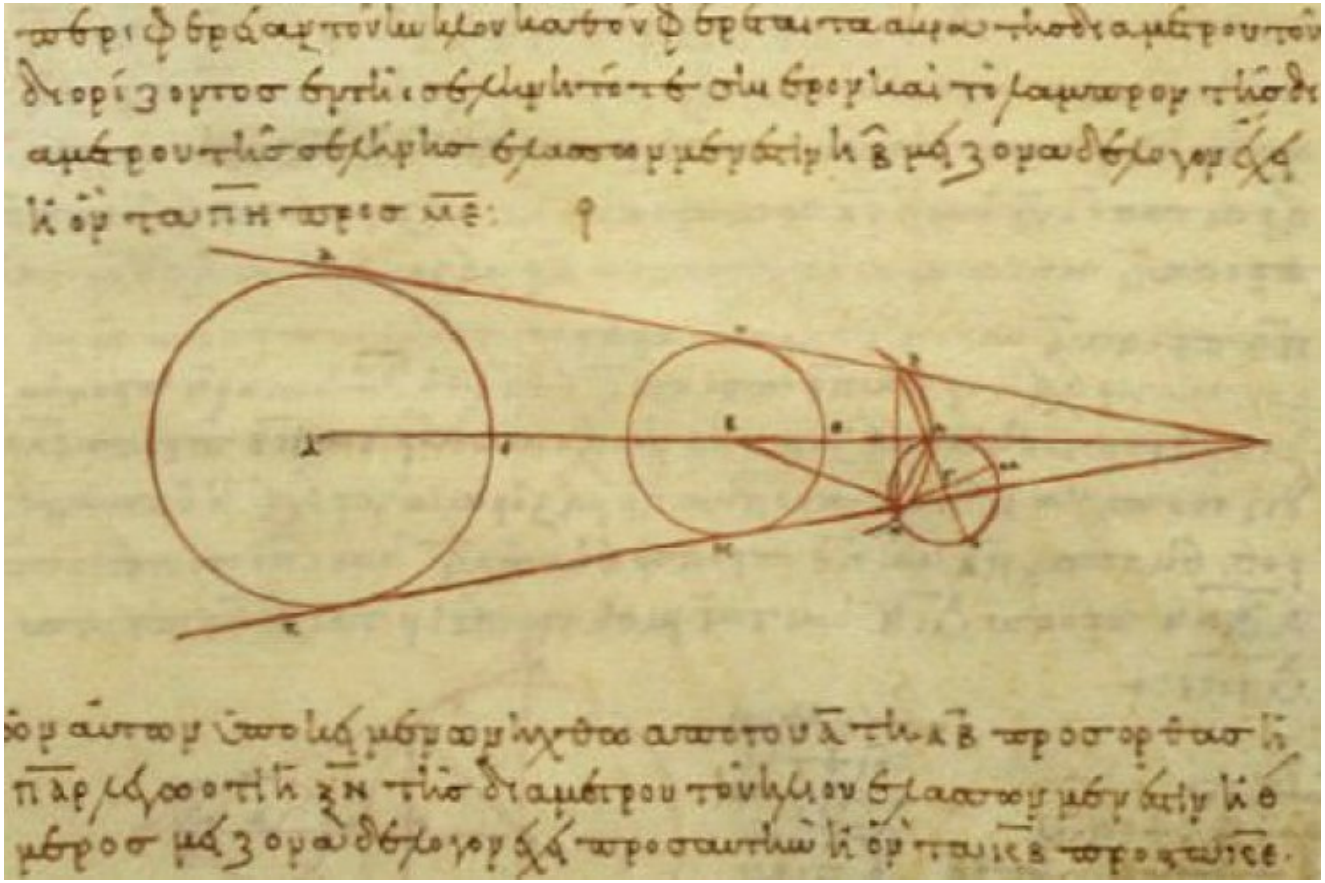
Babylonian clay tablet, with geometric and algebraic inscriptions.
Tell al Dhabba'i, Iraq, 2003-1595BC, Iraq Museum.



Clay tablet YBC 7289, displaying an approximation of the square root of 2 in four sexagesimal figures, 1 24 51 10, accurate to about six decimal digits.

$1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3} = 1.41421296...$

The tablet also gives an example where one side of the square is 30, and the resulting diagonal is 42 25 35 or 42.4263888...



10th century CE Greek copy of Aristarchus of Samos's (c.310 – c.230 BC) calculations of the relative sizes of the sun, moon and the earth.
Konstable, Wikimedia Commons.

Problems

Even after several passes at checking results, there might still be mistakes!

The amount of calculations that we are willing to perform by hand is quite limited.

Calculations

In 1588, Pietro Cataldi proved that

$$2^{17} - 1 = 131,071 \qquad \text{and} \qquad 2^{19} - 1 = 524,287$$

are both primes.

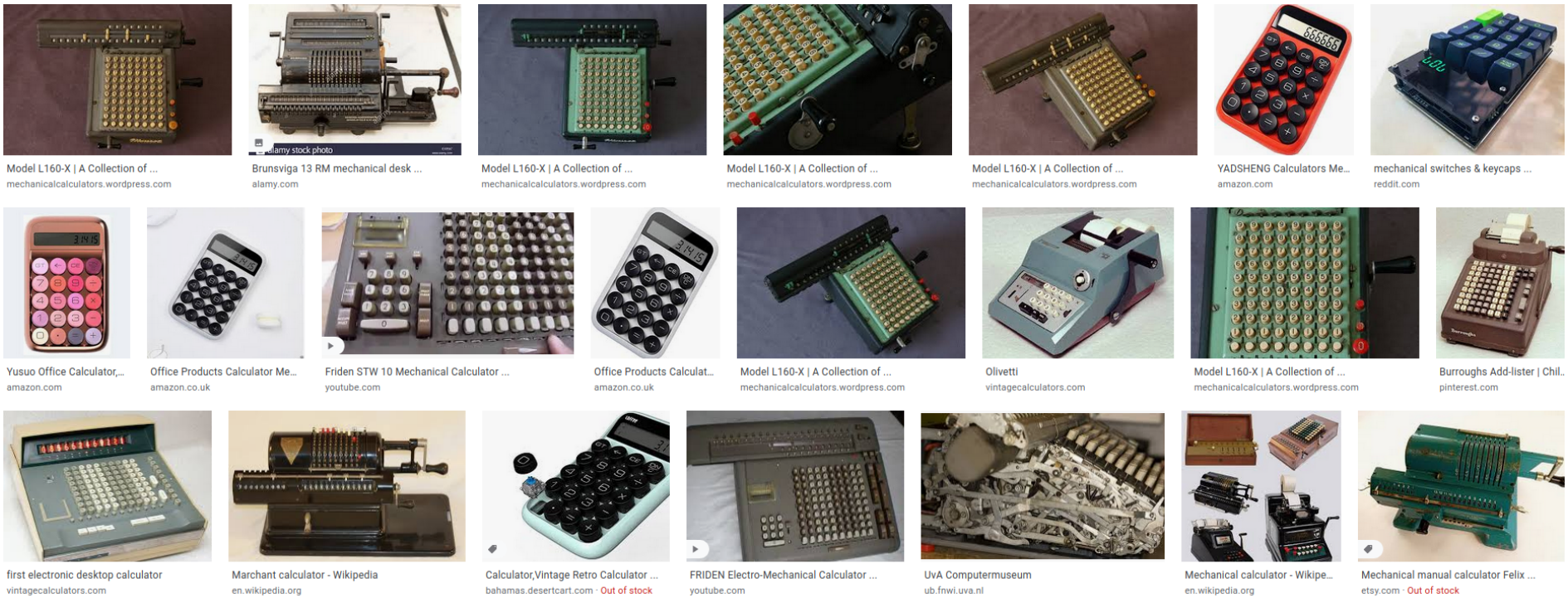
In 1772, Euler proved that $2^{31} - 1 = 2,147,483,647$ is prime.

In 1876, Lucas proved that
$2^{127} - 1 = 170,141,183,460,469,231,731,687,303,715,884,105,727$
is prime.

Source: The Largest Known prime by Year: A Brief History

Mechanical desk calculators

Enter the mechanical desk calculator...



The result of an image search for "mechanical desk calculators on Google"

In 1951 Ferrier proved the primality of $\frac{2^{148}+1}{17} =$

$$20,988,936,657,440,586,486,151,264,256,610,222,593,863,921.$$

Year	Discoverer	Prime
1588	Cataldi	$2^{17}-1 = 131,071$
1588	Cataldi	$2^{19}-1 = 524,287$
1772	Euler	$2^{31}-1 = 2,147,483,647$
1876	Lucas	$2^{127}-1 =$ 170,141,183,460,469,231,731,687,303,715,884,105,727
1951	Ferrier	$\frac{2^{148}+1}{17} =$ 20,988,936,657,440,586,486,151,264,256,610,222,593,863,921

The largest known prime in 2018 was found by Pace (GIMPS):

$$2^{77,232,917}-1 = \text{ a number with } 23,249,425 \text{ digits.}$$

Lucas's prime has 39 digits, Ferrier's prime has 44 digits.

With the help from computers

The largest known prime in 2018:

$$2^{77,232,917} - 1 = \text{ a number with } 23,249,425 \text{ digits.}$$

Today, it is hard to say even the *number of digits* of the largest known prime!

Regardless of whether finding large primes is mainstream mathematics, computers have an amazing potential for helping us doing mathematics.

What can be done with a computer?

Of course, this depends on how you program your computer!

Computers

- perform quickly vast numbers of simple operations;
- do not get bored by performing repetitive tasks;
- make surprisingly few mistakes;
- do not complain that they will not do your calculations, because there is no point to them.

Examples of tasks that you can expect a computer to perform faster and more accurately than a human:

- brute force searches,
- case bashes,
- tedious arithmetic operations,
- computing real numbers to high precision,
- finding yet another prime number...

Let us see how a computer can help us to prove that $524,287$ is prime.

Proving that $524,287$ is prime

Step 1. [**Human**] Let $n \in \mathbb{N}$ be a natural number.
n is prime if it does not have a factor in the range $\{2, 3, \ldots, \lfloor \sqrt{n} \rfloor\}$.

Step 2. [**Computer**] Evaluate $\sqrt{n}$ to at least one decimal digit.

$$\text{A: } \sqrt{524,287} \simeq 724.07665...$$

Step 3. [**Computer**] Divide $524,287$ by each integer

$$i \in \left\{2, 3, \ldots, 724 = \lfloor \sqrt{524,287} \rfloor\right\}.$$

A: The remainder 0 does not appear in the list.

Step 4. [**Human**] Conclude that $524,287$ is prime!

In this example, we used that the computer

- performs basic arithmetic operations;
- executes `loops`;
- accepts "`IF [...], THEN [...], ELSE [...]`" statements.

We formulate precise, simple, repetitive tasks.
The computer performs these tasks for ourselves.

We proved one mathematically interesting fact:

A number n is prime if it has no factors of size at most $\sqrt{n}$.

The computer takes care of the tedious part of the argument.

Performing 724 divisions of natural numbers with fewer than 10 digits is a breeze for a computer.

Hence, it could check that $n = 524,287$ is prime.

In theory, the same strategy works for every natural number:
the *theorem* we proved applies to *all* natural numbers n.

In practice, if we start with a really large number, then we may have to wait for a long time before we know the answer.

If we tried this strategy with $2^{77,232,917} - 1$ and we imagine that each particle in the universe is going to perform one division for us, we would need more than 10,000 universes to get our answers!

And then we need to search all these answers to see if there is a 0!

What else can we do with computers?

We can write better and more sophisticated algorithms using advanced programs, such as

Mathematica, MATLAB, Sage, PARI/GP,

Magma, Macaulay, SnapPy,...

At the same time, hardware development provides more computational power and allows us to reach further.

With such programs we can

- define variables,
- do symbolic computations,
- find Gröbner bases of ideals,
- approximate infinite series,
- estimate definite integrals,
- find presentations of groups,
- compute HOMFLY polynomials,
- ...

Experiments in mathematics

Suddenly, we can now really run experiments in mathematics: maths is an experimental science!

Integrated use of computers in the "creative" phase of mathematics means that you come up with ideas and the computer checks them!

For instance, imagine that we want to find a formula producing infinitely many primes. We better look for odd numbers, of course.

How about successors of powers of 2?

Primality of $2^a + 1$

- $2^0 + 1 = 2$: prime.
- $2^1 + 1 = 3$: prime.
- $2^2 + 1 = 5$: prime. Looking good! One step further...
- $2^3 + 1 = 9 = 3^2$: **not** prime.

Ok, we had a good run! Just our of curiosity, let us keep going.

- $2^4 + 1 = 17$: prime.
- $2^5 + 1 = 33 = 3 \cdot 11$: **not** prime.
- $2^6 + 1 = 65 = 5 \cdot 13$: **not** prime.
- $2^7 + 1 = 129 = 3 \cdot 43$: **not** prime.
- $2^8 + 1 = 257$: prime!

For $a \in \{1, 2, 4, 8\}$ we obtained that the number $2^a + 1$ is prime.
It seems that if a is a power of 2, then $2^a + 1$ is prime.

The next power of 2 after 8 is 16, so... is $2^{16} + 1$ prime?

$$2^{16} + 1 = 65,537$$

and it might require effort to check whether this is prime.
However, $2^{16} + 1$ is smaller than $2^{17} - 1$ and we saw that in 1588 Cataldi discovered that $2^{17} - 1$ is prime!

- $2^{16} + 1 = 65,537$ is prime!

These calculations suggest two separate statements.

Statement 1. If a **is not** a power of 2, then $2^a + 1$ is not prime.

Statement 2. If $a = 2^n$ **is** a power of 2, then $2^a + 1 = 2^{2^n} + 1$ is prime.

Statement 1. If a **is not** a power of 2, then $2^a + 1$ is not prime.

Statement 2. The integers of the form $2^{2^n} + 1$ are primes.

At the time of Fermat (1607–1665), one of these two statements was known to be true (and it is a fun exercise to prove it yourself!).

Fermat conjectured that the other was also true!

In 1732, Euler showed that $2^{2^5} + 1 = 2^{32} + 1 = 4,294,967,297$ is not prime, since 641 divides it.

As of today, the only known primes in the sequence $\left\{2^{2^n}\right\}$ are

$$2^1 + 1 \qquad 2^2 + 1 \qquad 2^4 + 1 \qquad 2^8 + 1 \qquad 2^{16} + 1.$$

They are called *Fermat primes*.

What next?

This is all fun! At least, I find it fun.

Using a computer, we can back up or find counterexamples to mathematical statements.

More complicated software and faster hardware will allow us to "get stuck" later on.

What else can computers do for us?

Long proofs

Some theorems in mathematics have incredibly long proofs.

A famous example is the Four Colour Theorem (Appel-Haken, 1976).

Theorem. The vertices of every planar graph can be coloured by at most 4 colours, so that adjacent vertices have different colours.

The proof involves isolating a finite, yet huge, set of planar graphs with the property that if these finitely many graphs are 4-colourable, then all planar graphs are 4-colourable.

This final statement has been checked by a computer.

Some mathematicians are not convinced by this result, since the computer might make mistakes.

The underlying assumption being that humans are infallible...?

Four Colour Theorem

Computers verified the outstanding cases of the Four Colour Theorem.

Even more: there is a computer program that checks the *logical structure* of the proof of the Four Colour Theorem!

Computer programs that verify the logical structure of mathematical proofs are called *proof assistants*.

Proof Assistants and Formalization

This is a step forward: there are computer programs that "know"

- the logical inference rules, (e.g. $P \wedge Q \Longrightarrow P$);
- the syntax of mathematical formulas,
  (e.g. $\forall n,\, a,\, b,\, c \in \mathbb{N} \setminus \{0\},\; a^n + b^n = c^n \Rightarrow n \leq 2$);
- a list of axioms, (e.g. the Zermelo-Fraenkel Axioms of Set Theory).

Using these rules, we can communicate to the computer

- the statement of a mathematical theorem,
- the steps of the proof, and
- the deduction rules used at each step.

The computer then verifies for us that our proof is correct!!

Several mathematical results have been formalized:

- the Four Colour Theorem,
- the Odd-Order Theorem (a finite group of odd order is soluble),
- the Continuum Hypothesis (a result about the independence of one of the axiom of Set Theory from the remaining ones).

Besides these "cornerstone" results, most results of the standard undergraduate mathematical curriculum have been formalized:

theorems about sequences,
limits, series, integrals,
classical real analysis,
functional analysis,
point-set topology,
metric spaces,
commutative algebra,
Hilbert's basis theorem,
existence of transcendental numbers,
...

Future steps

What comes next?

An *open problem* is a statement for which currently there is no proof and that, hopefully, is a consequence of results that we already proved.

Many known techniques might make progress on the proof, and yet they may not reach deep enough to prove the statement.

A "well-trained" proof assistant can get started on a proof of the open problem and inform us when it reaches a point where it is stuck.

If we can supply a proof of this result, then, in a favourable case, the proof assistant might be able to complete the proof for us!

A plan

- Start with some Axioms (e.g. the ZF Axioms of Set Theory).
- Using the Axioms and the rules of logic, prove more results.
- Formalize these results and their proofs in a proof checker.
- Build a mathematical library of results that the computer knows.

Eventually, the computer will "know" as much mathematics as we do. Possibly even more!

There are several workable interfaces for carrying out this plan:

Agda, Coq, HOL, Isabelle, Lean, Mizar,...

Automation

Now is the time to try automating!

We ask the computer to combine known results *on its own* to deduce new, provable results!

If we are interested in a specific theorem, we provide the assumptions and the goals and ask the computer to find a "path of proofs" connecting the hypotheses to the results.

While this may appear a far away dream... it is already happening!

At the moment, not much "new" mathematics is computer generated.

Yet, the computer can manufacture proofs of certain easy statements automatically.

Lean

I have only used the proof assistant called Lean.
Here are a few stats about its mathematical library.

The words "`lemma`" and "`theorem`" appear almost $39,000$ times.

"`simp`" is an all-purpose technique to ask Lean to simplify proofs.

It has been used over $13,000$ times.
Over $7,000$ times, `simp` proved the result.

Chances are that, if you type a theorem in Lean, you will be able to prove it by writing `simp`[1].

[1] Disclaimer: this is an exaggeration!

Lean

src > data > nat > prime.lean

```
310
311 end min_fac
312
313 theorem exists_dvd_of_not_prime {n : ℕ} (n2 : 2 ≤ n) (np : ¬ prime n) :
314   ∃ m, m | n ∧ m ≠ 1 ∧ m ≠ n :=
315 (min_fac n, min_fac_dvd _, ne_of_gt (min_fac_prime (ne_of_gt n2)).one_lt,
316   ne_of_lt $ (not_prime_iff_min_fac_lt n2).1 np)
317
318 theorem exists_dvd_of_not_prime2 {n : ℕ} (n2 : 2 ≤ n) (np : ¬ prime n) :
319   ∃ m, m | n ∧ 2 ≤ m ∧ m < n :=
320 (min_fac n, min_fac_dvd _, (min_fac_prime (ne_of_gt n2)).two_le,
321   (not_prime_iff_min_fac_lt n2).1 np)
322
323 theorem exists_prime_and_dvd {n : ℕ} (n2 : 2 ≤ n) : ∃ p, prime p ∧ p | n :=
324 (min_fac n, min_fac_prime (ne_of_gt n2), min_fac_dvd _)
325
326 /-- Euclid's theorem. There exist infinitely many prime numbers.
327 Here given in the form: for every `n`, there exists a prime number `p ≥ n`. -/
328 theorem exists_infinite_primes (n : ℕ) : ∃ p, n ≤ p ∧ prime p :=
329 let p := min_fac (n! + 1) in
330 have f1 : n! + 1 ≠ 1, from ne_of_gt $ succ_lt_succ $ factorial_pos _,
331 have pp : prime p, from min_fac_prime f1,
332 have np : n ≤ p, from le_of_not_ge $ λ h,
333   have h₁ : p | n!, from dvd_factorial (min_fac_pos _) h,
334   have h₂ : p | 1, from (nat.dvd_add_iff_right h₁).2 (min_fac_dvd _),
335   pp.not_dvd_one h₂,
336 ⟨p, np, pp⟩
337
338 lemma prime.eq_two_or_odd {p : ℕ} (hp : prime p) : p = 2 ∨ p % 2 = 1 :=
339 (nat.mod_two_eq_zero_or_one p).elim
340   (λ h, or.inl ((hp.2 2 (dvd_of_mod_eq_zero h)).resolve_left dec_trivial).symm)
341   or.inr
342
343 theorem coprime_of_dvd {m n : ℕ} (H : ∀ k, prime k → k | m → ¬ k | n) : coprime m n :=
344 begin
345   cases eq_zero_or_pos (gcd m n) with g0 g1,
346   { rw [eq_zero_of_gcd_eq_zero_left g0, eq_zero_of_gcd_eq_zero_right g0] at H,
347     exfalso,
348     exact H 2 prime_two (dvd_zero _) (dvd_zero _) },
```

▼ prime.lean:336:0

▼ Tactic state

expected type:

```
n : ℕ
p : ℕ := (n! + 1).min_fac
f1 : n! + 1 ≠ 1
pp : prime p
np : n ≤ p
⊢ n ≤ p ∧ prime p
```

▶ All Messages (0)

What Euclid's proof of the infinitude of primes looks like in Lean

Thank you!

Questions?