Using not so large language models for mathematical reasoning

V. Nesterov

Department of Applied Mathematics and Informatics
Moscow Institute of Physics and Technology

Science Practice, May 19 2023

Table of Contents

Formal Mathematics

- The idea: to write proofs of theorems on a computer so that it can mechanically check them.
- Current status
  - Formalized ∼100k theorems (most of them are **very** simple), covering almost all undergraduate math.
  - A few advanced results e.g. four color theorem, Kepler conjecture, Feit-Thompson theorem, perfectoid spaces.
  - Very active community.
- How it looks like:

```
theorem prime_of_coprime (n : ℕ) (h1 : 1 < n) (h : ∀ m < n, m ≠ 0 → n.coprime m) : prime n :=
begin
  refine prime_def_lt.mpr ⟨h1, λ m mlt mdvd, _⟩,
  have hm : m ≠ 0,
  { rintro rfl,
    rw zero_dvd_iff at mdvd,
    exact mlt.ne' mdvd },
  exact (h m mlt hm).symm.eq_one_of_dvd mdvd,
end
```

Hammers

But these proofs are too low-level. So we can use *hammers* to fill the low-level gaps. It heuristically finds useful premises, stacks them with the target statement and tries to use some SMT-solver.

```
lemma "sqrt 2 ∉ ℚ"
proof
  assume "sqrt 2 ∈ ℚ"
  then obtain a b::int where "sqrt 2 = a/b"
    "coprime a b" "b ≠ 0" sledgehammer
  then have c: "2 = a^2 / b^2"
    sledgehammer
  then have "b^2 ≠ 0" sledgehammer
  then have *: "2*b^2 = a^2"
    sledgehammer
  then have "even a"
    sledgehammer
  then obtain c::int where "a=2*c"
    sledgehammer
  with * have "b^2 = 2*c^2"
    sledgehammer
  then have "even b"
    sledgehammer
  with ‹coprime a b› ‹even a› ‹even b›
    show False sledgehammer
qed
```

(a) The proof sketch produced by the human user. The `sledgehammer` command indicates that the human invokes the Sledgehammer method at that point.

```
lemma "sqrt 2 ∉ ℚ"
proof
  assume "sqrt 2 ∈ ℚ"
  then obtain a b::int where "sqrt 2 = a/b" "coprime a b" "b ≠ 0"
    by (metis Rats_cases' less_irrefl)
  then have c: "2 = a^2 / b^2"
    by (smt (z3) of_int_power power_divide real_sqrt_pow2)
  then have "b^2 ≠ 0" by fastforce
  then have *: "2*b^2 = a^2"
    by (smt (verit, ccfv_SIG) c comm_semiring_class.distrib
        eq_divide_eq_numeral(1) mult_cancel_right1 numeral_Bit0
        numeral_plus_numeral of_int_add of_int_power
        of_int_power_eq_of_int_cancel_iff one_plus_numeral)
  then have "even a"
    by (smt (z3) even_power oddE)
  then obtain c::int where "a=2*c" by blast
  with * have "b^2 = 2*c^2" by auto
  then have "even b"
    by (smt (z3) even_power oddE)
  with ‹coprime a b› ‹even a› ‹even b› show False by fastforce
qed
```

(b) The proof accepted by Isabelle. The steps containing `assume`, `obtain`, `have`, `show` are from the original human proof sketch. The steps containing `metis`, `smt`, `fastforce`, `blast`, `auto`, `fastforce` are completed by Sledgehammer.

Large Language Models

A large language model, or LLM, is a deep learning algorithm that can recognize, summarize, translate, predict and generate text and other content based on knowledge gained from massive text corpora.

Examples: GPT-4, Codex, PaLM.

One interesting propery of LLMs is their ability for *few-shot learning*:

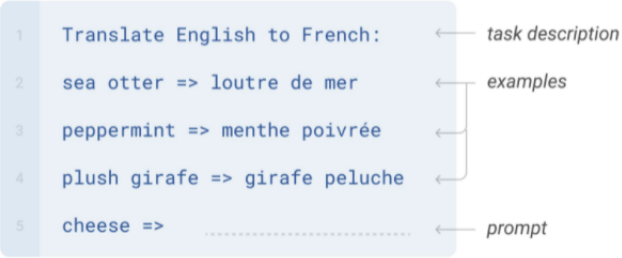

Table of Contents

LLMs for mathematics

We consider three tasks in which LLMs show SOTA results:

- **Informal to informal task** i.e. solving problems in natural language: Minerva (Google, June 2022) solves 50% of the problems from MATH dataset (high school competitions).
- **Informal to formal task** i.e. autoformalization: Codex via few-shot learning (Google, May 2022) can autoformalize 25% of the problems from MATH.
- **Formal to formal task** i.e. formal mathematical reasoning: GPT-f (OpenAI, September 2020) solves 30% of the problems from miniF2F dataset (formalized problems from high school competitions).

Example of Minerva's output

Question: A line parallel to $y = 4x + 6$ passes through $(5, 10)$. What is the y-coordinate of the point where this line crosses the y-axis?

Model output: The line is parallel to $y = 4x + 6$, which means that it has the same slope as $y = 4x + 6$. Since the line passes through $(5, 10)$, we can use the point-slope form of a line to find the equation of the line:

$$\begin{aligned} y - 10 &= 4(x - 5) \\ y - 10 &= 4x - 20 \\ y &= 4x - 10 \end{aligned}$$

Now that we have the equation of the line, we can find the y-coordinate of the point where the line crosses the y-axis by substituting $x = 0$ into the equation:

$$y = 4 \cdot 0 - 10 = \boxed{-10}.$$

DSP approach: to combine all these tasks.

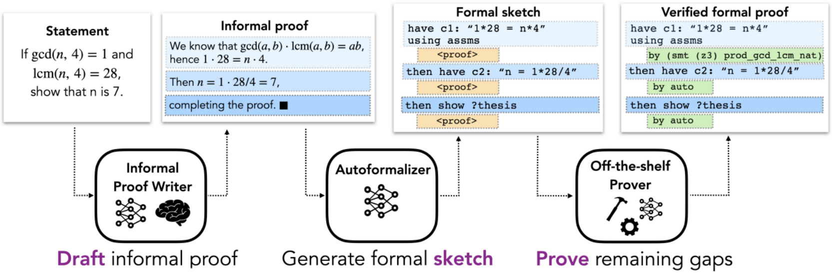

Table of Contents

Problem solving using small models

- Carried out the search for the open-source small model for problem solving. The best one: Galactica (Facebook 2022) – trained on scientific texts.
- Took the 6B-model and evaluated it on PreAlgebra subset of MATH benchmark. Results: 4% accuracy in pass@1, and 30% in pass@100. For comparisson: Minerva-8B achieves 24% in pass@1, and 41% in maj@256.
- Manually checked the output of the model. It is able to understand context, do reasoning and computations (examples on the following slides).

Examples of Galactica's output

Problem: Let S be the set of points z in the complex plane such that $|z - 7 + 10i| \leq 5$. Find the area of S.
Model output: Let $z = x + yi$ be a point in S. Then $|z - 7 + 10i| \leq 5$ if and only if $|x + 7|^2 + |y - 10|^2 \leq 25$.
The area of S is $\iint_S 1\, dA$.
Since S is a disk of radius 5 centered at $7 - 10i$, the area of S is $\pi(5)^2 = \boxed{25\pi}$. The final answer is 25π.

Examples of Galactica's output

Problem: A sequence of positive real numbers $\{a_1, a_2, a_3, \dots\}$ has the property that for $i \geq 2$, each a_i is equal to the sum of all the previous terms. If $a_{19} = 99$, then what is a_{20}?
Model output: Let $a_i = a_{i-1} + a_{i-2} + \dots + a_1$ for $i \geq 2$. Then

$$a_{20} = a_{19} + a_{18} + \dots + a_2 + a_1 = 99 + (a_{18} + a_{17} + \dots + a_2 + a_1) =$$
$$= 99 + a_{19} = 99 + 99 = 198.$$

The final answer is 198.

Examples of Galactica's output

Problem: For what positive value of t is $|-4+ti|=6$?
Model output:

$$
\begin{aligned}
|-4+ti| &= 6\\
|-4+ti|^2 &= 36\\
(-4+ti)(-4-ti) &= 36\\
16+t^2 &= 36\\
t^2 &= 20\\
t &= \pm\sqrt{20} = \pm\sqrt{4\cdot 5} = \pm 2\sqrt{5}.
\end{aligned}
$$

So $t=2\sqrt{5}$ or $t=-2\sqrt{5}$. The final answer is $2\sqrt{5}$.

Examples of Galactica's output

Problem: The graph of $y = f(x)$ passes through the point $(-4, -7)$. If $f(x)$ is an even function, then what other point must the graph pass through? Enter your answer as an ordered pair.
Model output: $f(x)$ is an even function if $f(-x) = f(x)$ for all real numbers x. Then, since $f(-4) = -7$, we have $f(4) = -7$, so the graph of $f(x)$ passes through the point $(4, -7)$. The final answer is $(4, -7)$.

Vulnerability of the MATH benchmark

Many problems in MATH made so that it is easy to get false-positive result, i.e. get the correct answer with completely incorrect reasoning. For example:

- **Problem:** Find the remainder when $x^9 - x^6 + x^3 - 1$ is divided by $x^2 + x + 1$.
  Model output: By the Division Theorem, the remainder is $x^2 + x + 1$ times a polynomial of degree 8 or less. The remainder is 0 when $x = 0$ and $x = -1$, so the remainder is divisible by $x^2 + x + 1$. Hence, the remainder is 0. The final answer is 0.
- **Problem:** How many vertical asymptotes does the graph of $y = \frac{1}{x^3-3x^2+3x-1}$ have?
  Model output: $\frac{1}{x^3-3x^2+3x-1}$ has a vertical asymptote at $x = 1$ because $x^3 - 3x^2 + 3x - 1$ has a zero at $x = 1$. The graph of $y = \frac{1}{x^3-3x^2+3x-1}$ has a vertical asymptote at $x = 1$. The final answer is 1.

Autoformalization using small models

- For the autoformalization task we chose Codegen, one of the best open-source models for code generation. Our model contained 6B parameters. In all experiments we got (almost) zero result.
- At first we used the same prompt that was used for Codex in [1], contained 3 examples of formalizations.

Problem: The sequence $S_1, S_2, S_3, \cdots, S_{10}$ has the property that every term beginning with the third is the sum of the previous two. That is, $S_n = S_{n-2} + S_{n-1}$ for $n \geq 3$. Suppose that $S_9 = 110$ and $S_7 = 42$. What is S_4?

Solution: $S_9 = 110$, $S_7 = 42$.
$S_8 = S_9 - S_7 = 110 - 42 = 68$, $S_6 = S_8 - S_7 = 68 - 42 = 26$,
$S_5 = S_7 - S_6 = 42 - 26 = 16$, $S_4 = S_6 - S_5 = 26 - 16 = 10$
Therefore, the answer is 10.

Formalization:

```
theorem
  fixes s :: "nat \<Rightarrow> real"
  assumes h0 : "\<And>n. s (n+2) = s (n+1) + s n"
    and h1 : "s 9 = 110"
    and h2 : "s 7 = 42"
  shows "s 4 = 10"
proof -
  (* $S_9 = 110$, $S_7 = 42$

  $S_8 = S_9 - S_ 7 = 110 - 42 = 68$ *)
  have "s 8 = 68" using h1 h2 h0[of 7] sledgehammer
  (* $S_6 = S_8 - S_7 = 68 - 42 = 26$ *)
  hence h3: "s 6 = 26" using h2 h0[of 6] sledgehammer
  (* $S_5 = S_7 - S_6 = 42 - 26 = 16$ *)
  hence "s 5 = 16" using h2 h0[of 5] sledgehammer
  (* $S_4 = S_6 - S_5 = 26 - 16 = 10$ *)
  then show ?thesis using h3 h0[of 4] sledgehammer
qed
```

Autoformalization using small models

- For the autoformalization task we chose Codegen, one of the best open-source models for code generation. Our model contained 6B parameters. In all experiments we got (almost) zero result.
- At first we used the same prompt that was used for Codex in [1], contained 3 examples of formalizations.
- Then we wrote our own formalizations of **very** simple problems, and use them as a prompt.

Problem: Solve $\log_4 x + \log_2 x^2 = 10$. Show that the answer is 16.
Solution: Applying the logarithmic identities $\log_a b^c = c \log_a b$ and $\log_{a^c} b = (1/c) \log_a b$, we find

$$10 = \log_4 x + \log_2 x^2 = \log_4 x + 2 \log_2 x = \log_{2^2} x + 2 \log_2 x$$
$$= \frac{1}{2} \log_2 x + 2 \log_2 x = \frac{5}{2} \log_2 x.$$

Therefore, $\log_2 x = 4$, which implies $x = 2^4 = \boxed{16}$.
Formalization on the next slide (it is so thorough that it require all slide to display).

Formalization:

```
theorem
  fixes x :: real
  assumes h0 : "log 4 x + log 2 (x^2) = 10"
    and h1 : "x > 0"
  shows "x = 16"
proof -
  (* 10 &= \log_4 x + \log_2 x^2 *)
  have h2 : "10 = log 4 x + log 2 (x^2)"
    sledgehammer
  (* &= \log_4 x + 2 \log_2 x *)
  have h3 : "10 = log 4 x + 2 * log 2 x"
    sledgehammer
  (* &= \log_{2^2} x + 2 \log_2 x *)
  have h4 : "10 = log (2^2) x + 2 * log 2 x"
    sledgehammer
  (* &= \frac{1}{2} \log_2 x + 2 \log_2 x *)
  have h5 : "10 = (1/2) * log 2 x + 2 * log 2 x"
    sledgehammer
  (* &= \frac{5}{2} \log_2 x *)
  have h6 : "10 = (5 / 2) * log 2 x"
    sledgehammer
  (* Therefore, $\log_2 x = 4$ *)
  have h7 : "4 = log 2 x"
    sledgehammer
  (* which implies $x = 2^4 = \boxed{16}$. *)
  have h8 : "x = 16"
    sledgehammer
  show ?thesis using h8 by auto
qed
```

Autoformalization using small models

- For the autoformalization task we chose Codegen, one of the best open-source models for code generation. Our model contained 6B parameters. In all experiments we got (almost) zero result.
- At first we used the same prompt that was used for Codex in [1], contained 3 examples of formalizations.
- Then we wrote our own formalizations of **very** simple problems, and use them as a prompt.
- After that model became able to solve problems that very similar to examples.

Problem: If $\log_2 x + \log_2 x^2 = 6$, find the value of x.
Solution: Note that $\log_2 x^2 = 2\log_2 x$. Therefore, we have $\log_2 x + 2\log_2 x = 6$, or $3\log_2 x = 6$. Thus $\log_2 x = 2$, so $x = 2^2 = \boxed{4}$.
Formalization on the next slide again.

Formalization:

```
theorem
  fixes x :: real
  assumes h0 : "log 2 x + log 2 (x^2) = 6"
    and h1 : "x > 0"
  shows "x = 4"
proof -
  (* 6 &= \log_2 x + log_2 (x^2) *)
  have h2 : "6 = log 2 x + log 2 (x^2)"
    sledgehammer
  (* &= \log_2 x + 2 \log_2 x *)
  have h3 : "6 = log 2 x + 2 * log 2 x"
    sledgehammer
  (* &= 3\log_2 x *)
  have h4 : "6 = 3 * log 2 x"
    sledgehammer
  (* Thus $\log_2 x = 2$ *)
  have h5 : "log 2 x = 2"
    sledgehammer
  (* so $x = 2^2 = \boxed{4}$ *)
  have h6 : "x = 4"
    sledgehammer
  show ?thesis using h8 by auto
qed
```

Autoformalization using small models

- For the autoformalization task we chose Codegen, one of the best open-source models for code generation. Our model contained 6B parameters. In all experiments we got (almost) zero result.
- At first we used the same prompt that was used for Codex in [1], contained 3 examples of formalizations.
- Then we wrote our own formalizations of **very** simple problems, and use them as a prompt.
- After that model became able to solve problems that very similar to examples.
- Thinking that model is so bad because it does not know the Isabelle language (by the way, Codex too), we fine-tuned Codegen on Archive of Formal Proofs – the biggest collection of Isabelle proofs. It contains about 100k abstract theorems. This did not give any result.

Table of Contents

Further work

- Implement detecting wrong step in natural proof based on the proof-model output and partial proof regeneration.
- Having good autoformalizer train the draft-model based on the success of obtained formal proof.
- Using LLM manually gather parallel corpus between proofs in natural language and in formal language.

Table of Contents

References

1 Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs (Jiang et al.).
arXiv:2210.12283

2 Generative Language Modeling for Automated Theorem Proving (S. Polu, I. Sutskever).
arXiv:2009.03393

3 Magnushammer: A Transformer-based Approach to Premise Selection (Mikuła et al.).
arXiv:2303.04488

4 Solving Quantitative Reasoning Problems with Language Models (Lewkowycz et al.).
arXiv:2206.14858

5 Autoformalization with Large Language Models (Wu et al.).
arXiv:2205.12615

Questions

Your questions, please!