Leveraging Language Models for Autoformalizing Theorems: A Case Study

First Author[1[0000−1111−2222−3333]] and Second Author[2[1111−2222−3333−4444]]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com
http://www.springer.com/gp/computer-science/lncs

1 Introduction

Large Language Models (LLMs) like Mistral have garnered considerable attention for their potential to automate various tasks in formal theorem proving. However, their adoption in this context is hindered by challenges such as inaccuracies, contextual understanding gaps, comprehending mathematical concepts, resolving natural language ambiguities and handling complex proofs. In recent work [4], strategies have been explored to enhance LLMs for proof generation, including but not limited to providing access to proof states and file dependencies, utilizing error-based learning, and enhancing diversity through prompt engineering.

In this work we leverage the capabilities of Mistral for autoformalization in mathematics in conjuction with the Lean 4 theorem prover [3] and its mathematical library `mathlib` [1]. More specifically we aim to autoformalize theorems in transcedental number theory with formal proof sketches being the main result. First, we preprocess the text to extract relevant mathematical statements from the target source. Next, we fine-tune Mistral on these to enhance its understanding of mathematical concepts and language. We then use prompt engineering to formalize the extracted statements from the book, excluding proofs. The final step would be to generate formal proofs using the formalized statements.

2 Enhancing Mistral's Performance: Insights and Strategies

During few-shot prompting, challenges were addressed through strategies such as clear mathematical notation, problem decomposition, providing context like definitions and theorems, adhering to Lean 4 syntax, using examples, and iterative testing of generated code, resulting in improved model performance. A centralized prompt repository[3]. ensuring consistency through meticulous tracking of modifications is provided.

[3] https://github.com/mkaratarakis/autoformalization-LLMs

Autoformalization revealed issues like undefined or misused variables, rectified by removing unnecessary variables and adjusting definitions. Errors were corrected by replacing incorrect statements with the `sorry` identifier – a command that produces a proof of anything or provides an object of any data type – and adding missing variable declarations to ensure accuracy, emphasizing mathematical understanding.

In our investigation, defining necessary concepts and prerequisite theorems posed challenges, requiring guidance and modification to align with established conventions and fit within the `mathlib` framework. Once correct definitions were established, the partial formalization became more accurate, highlighting the effectiveness of meticulous attention to definitions and prerequisites.

2.1 Strategies for Performance Improvement

Enhancing Mistral's performance involves a multifaceted approach. This includes evaluating its mathematical comprehension and gradually providing additional context to aid its understanding. Employing incremental prompting for subproofs and computations further augments Mistral's ability to generate accurate formalizations.

2.2 Mistral's Reliance on Lean 3 Syntax

Mistral's training data, limited to Lean 3 syntax until January 2022, influences its usage of syntax. To address this, tailored instructions are necessary, entailing modifications to function notation and syntax adjustments. Moreover, instructions to exclude outdated imports are imperative, recognizing the transition to Lean 4 as the standard version.

2.3 Handling Type Mismatch Errors

Type mismatch errors, frequently encountered in theorem provers like Lean 4, signify discrepancies between expected and actual types. Resolving these errors entails a thorough logic review and alignment with expected types to ensure consistency and accuracy in formalizations.

2.4 Addressing Autonomy Challenges

In such cases, the challenge lies in the LLM generating its own prerequisite definitions and lemmata that are not present in the library. The customary approach addresses this challenge by supplying the LLM with definitions from existing libraries like `mathlib` or with auxiliary definitions defined by the used. This practice not only reduces the LLM's failure rate but also ensures consistency and enhancing efficiency.

2.5 Addressing Prioritization Issues

Mistral's prioritization of prerequisite theorems or definitions over the main task requires careful validation of definitions. Users must ensure that proofs address the intended problem, avoiding distractions with unrelated lemmata or definitions. This validation involves reviewing proof steps, checking relevance to the original problem, and potentially adjusting the theorem statement or proof strategy to guide the prover effectively.

References

1. mathlib Community.: The Lean mathematical library. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020). pp. 367–381 (2020)
2. Hua, L.K.: Introduction to number theory. Springer Science & Business Media (2012)
3. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The Lean Theorem Prover (System Description). In: Felty, A.P., Middeldorp, A. (eds.) Automated Deduction - CADE-25. pp. 378–388. Springer International Publishing, Cham (2015)
4. Zhang, S.D., Ringer, T., First, E.: Getting more out of large language models for proofs (2023)

3 Appendix

We showcase the prerequisite definitions, informal statements and proofs for two lemmata (p. 489-490, [2]), alongside the proof sketches obtained through our Mistral experiments.

Example 1

Theorem 31 (Lemma 8.1 ([2]) *Let $0 < M < N$, and a_{jk} be rational integers satisfying $|a_{jk}| \leq A$ where $1 \leq A$, $1 \leq j \leq M$ and $1 \leq k \leq N$. Then there exists a set of rational integers $x_1 ..., x_N$, not all zero, satisfying $a_{j1}x_1 + \cdots a_{jN}x_N = 0,\ 1 \leq j \leq M$ and $|x_k| \leq (NA)^{\frac{M}{N-M}},\ 1 \leq k \leq N$.*

Proof. Let $H = (NA)^{(\frac{M}{N-M})}$. Then $NA < (H+1)^{(\frac{N-M}{M})}$.

Hence $(NAH) + 1 \leq NA(H+1)$ and $NA(H+1) < (H+1)^{\frac{N}{M}}$

Define

$$y_j = a_{j1}x_1 + \cdots a_{jN}x_N,\ 1 \leq j \leq M. \tag{1}$$

We define B_j as the sum of the $-min(0, a_{jk})$ for all a_{jk}.

Similarly, we define C_j as the sum of the $max(0, a_{jk})$ for all a_{jk}.

For any set of integers $(x_1, \ldots, x_N)$ satisfying

$$0 \leq x_k \leq H,\ 1 \leq k \leq N. \tag{2}$$

we have that $-B_jH \leq y_j \leq C_jH$, and $B_j + C_j \leq NA$.
The number of sets of $(x_1, \ldots, x_N)$ satisfying

$$0 \leq x_k \leq H, \ 1 \leq k \leq N. \tag{3}$$

is $(H+1)^N$.
And the corresponding number of set of sets $(y_1, \ldots, y_M)$ is at most

$$(NAH+1)^M.$$

It follows from the fact

$$(NAH) + 1 \leq NA(H+1) < (H+1)^{\frac{N}{M}} \tag{4}$$

and the pigeonhole principle that there must be two sets $(x'_1, \ldots, x'_N)$ and $(x''_1, \ldots, x''_N)$ which correspond to the same set $(y_1, \ldots, y_M)$.
Let $x_k = x'_k - x''_k, (1 \leq k \leq N)$ so that $(x_1, \ldots, x_N)$ is now the required set satisfying

$$a_{j1}x_1 + \cdots a_{jN}x_N = 0, \ 1 \leq j \leq M \tag{5}$$

and

$$|x_k| \leq (NA)^{\frac{M}{N-M}}, 1 \leq k \leq N. \tag{6}$$

After multiple iterations and asking the system to remove the `let` statements and change the notation, nothing changed and the system was stuck into giving the same answer over and over again. Thus, we started a new chat, included the statement we had in Lean 4 code and asked the system to reformalize the partial proof.

After several prompts, the system successfully produced the correct syntax. Notably, Mistral employed conventional mathematical notation, such as # for the cardinality of sets, which diverges from Lean's syntax.

Through further iterations, we achieved a stage where the auto-formalization of both the primary statement and its sub-proof statements became reasonably accurate. However, despite our continuous efforts to prompt, we were unable to ensure that the LLM adhered to Lean4 syntax, including the replacement of `begin` with `by` and the elimination of `end` commands.

```
-- Prove that for any set of integers x_k satisfying 0 ≤ x_k ≤
  H,
-- -B_j*H ≤ y_j ≤ C_j*H and B_j + C_j ≤ N*A
have hYBound : ∀ (x : Fin N → ℤ), (∀ k, 0 ≤ x k ∧ x k ≤ H) →
  ∀ j, -B j * H ≤ y j ∧ y j ≤ C j * H ∧ B j + C j ≤ N * A :=
begin
  intros x hx
  intros j
  -- Prove -B_j*H ≤ y_j
  have hYL : -B j * H ≤ y j := sorry
  -- Prove y_j ≤ C_j*H
```

```
    have hYR : y j ≤ C j * H := sorry
    -- Prove B_j + C_j ≤ N*A
    have hBC : B j + C j ≤ N * A := sorry
    exact ⟨hYL, hYR, hBC⟩
  end

  -- Prove that the number of sets of x_k is (H + 1)^N
  have hXCount : # X = (H + 1) ^ N := sorry

  -- Prove that the number of corresponding sets of y_j is at
    most (NAH + 1)^M
  have hYCount : # Y ≤ (N * A * H + 1) ^ M :=
  begin
    -- The proof is left as "sorry" for now
    sorry
  end

  -- Apply the pigeonhole principle to find two sets of x_k that
    correspond to the same set of y_j
  have hPigeonhole : ∃ x′ x″ ∈ X, ∀ j, y j = ∑ k, a j k * x′ k ∧
    y j = ∑ k, a j k * x″ k :=
  begin
    -- The proof is left as "sorry" for now
    sorry
  end

  -- Define x_k as the difference between the two sets of x_k
  let x : Fin N → ℤ := fun k => x′ k - x″ k

  -- Prove that x_k satisfies the required conditions
  have hX : (∃ k, x k ≠ 0) ∧ ∀ j, ∑ k, a j k * x k = 0 ∧ ∀ k, |
    x k| ≤ (N * A)^(M / (N - M)) :=
  begin
    -- The proof is left as "sorry" for now
    sorry
  end

  -- Complete the proof by existential introduction
  exact ⟨x, hX⟩
end
```

Example 2

The subsequent step in our process involves establishing a definition, a task acknowledged as one of the most challenging aspect in formalization endeavors. Presently, our objective is to assist Mistral in formulating the definition of the house of an algebraic integer. Following the initial prompt, the expected response follows a familiar pattern, with Mistral attempting independently to delineate the fundamental prerequisites but without achieving success.

```
-- Define the conjugates of an algebraic integer
def conjugates (α : K) : List K :=
  -- insert definition here

-- Define the house of an algebraic integer
def house (α : K) : ℝ :=
  (conjugates α).map (fun x => abs x).max′
```

Additionally, Mistral encounters difficulty in completing specific definitions, such as `conjugates`. The customary approach in such cases involves furnishing the LLM with foundational definitions that it cannot generate autonomously but are already available in `mathlib`. A few more prompts eventually led the LLM to propose the following definition:

```
-- Define the house of an algebraic integer
def house (α : K) : ℝ :=
  let embeddings :=
    NumberField.Embeddings.range_eval_eq_rootSet_minpoly α
  (embeddings.map (fun x => abs x)).max′
```

Subsequently, efforts were directed towards rectifying this definition, and subsequent prompts were formulated accordingly. Following several prompts, a partial formalization was successfully attained.

```
variable {K : Type*} [Field K] [NumberField K]

instance :  Algebra ℚ K := sorry

-- Define a theorem stating that the set of absolute values of
    the conjugates of an algebraic integer is nonempty
theorem nonempty_conjugates_abs (α : K) :
  (Set.toFinset (Set.image Complex.abs (Polynomial.rootSet
    (minpoly ℚ α) ℂ))).Nonempty :=
sorry

-- Define the house of an algebraic integer
noncomputable def house (α : K) : ℝ :=
  Finset.max′ (Set.toFinset (Set.image Complex.abs
    (Polynomial.rootSet (minpoly ℚ α) ℂ)))
  (nonempty_conjugates_abs α)
```

Example 3

The next theorem we aim to address is as follows:

Theorem 32 (Lemma 8.2 [2]) *Let $0 < p < q$, and a_{kl} be rational integers satisfying $\overline{|a_{kl}|} \leq A$ where $A \geq 1$, $1 \leq k \leq p$ and $1 \leq l \leq q$. Then there exists a set of rational integers $\xi_1 \dots, \xi_q$, not all zero, satisfying*

$$a_{k1}\xi_1 + \cdots + a_{kq}\xi_q = 0, \quad 1 \leq k \leq p, \quad 1 \leq l \leq q. \tag{7}$$

and

$$\overline{|\xi_l|} < c_1(1 + (c_1 qA)^{\frac{p}{q-p}}) \tag{8}$$

Proof. Let

$$\xi_l = x_{l1}\beta_1 + \cdots + x_{lh}\beta_h, \quad (1 \leq l \leq q) \tag{9}$$

where $x_{1l}, \ldots, x_{lh}$ are rational integers.

Let $a_{kl}\beta_r = a_{klr1}\beta_1 + \cdots a_{klrh}\beta_h$

where $a_{klr1}, \ldots, a_{klrh}$ are also rational integers. For $1 \leq k \leq p$, we have, from $a_{k1}\xi_1 + \cdots a_{kq}\xi_q = 0,\ 1 \leq k \leq p\ ,\ 1 \leq l \leq q$, that

$$\begin{aligned} 0 &= \sum_{l=1}^{q} a_{kl}\xi_l & (10)\\ &= \sum_{n=1}^{q} a_{kl} \sum_{r=1}^{h} x_{lr}\beta_r & (11)\\ &= \sum_{n=1}^{h} \sum_{l=1}^{q} x_{lr} \sum_{r=1}^{h} a_{aklru}\beta_u & (12)\\ &= \sum_{u=1}^{h} \left(\sum_{r=1}^{h} \sum_{l=1}^{q} a_{klru}x_{lr} \right) \beta_u & (13) \end{aligned}$$

Since $\beta_1, \ldots, \beta_h$ are linearly independent we have the hp number of equations

$$\sum_{r=1}^{h} \sum_{l=1}^{q} a_{klru}x_{lr}, \quad 1 \leq u \leq h, \quad 1 \leq k \leq p \tag{14}$$

with hq number of unknowns.

From $a_{kl}\beta_r = a_{klr1}\beta_1 + \cdots a_{klrh}\beta_h$, and our remark preceeding Lemma 8.1 we see that

$$\begin{aligned} \overline{\mid a_{klru} \mid} &\leq c \max_{1 \leq i \leq h} \beta^{(i)} A & (15)\\ &\leq c_2 A. & (16) \end{aligned}$$

It now follows from Lemma 8.1 that the system (8) has a non trivial set of solutions in rational integers satisfying

$$\mid x_{lr} \mid \leq 1 + (hqc_2A)^{\frac{p}{(p-q)}}, \quad 1 \leq l \leq q \text{ and } 1 \leq r \leq h. \tag{17}$$

Therefore

$$\begin{aligned} \overline{\mid \xi_l \mid} &\leq \mid x_{l1} \mid \overline{\mid \beta_1 \mid} + \cdots + \mid x_{lh} \mid \overline{\mid \beta_h \mid} \\ &\leq c_2 h (1 + (hqc_2A)^{\frac{p}{(q-p)}}). & (18) \end{aligned}$$

Taking $c_1 = c_2 h$ the lemma is proved.

After some prompting we managed to get the statement of the theorem almost correct.

```
theorem lemma_8_2 (p q : ℕ) (hp : 0 < p) (hq : 0 < q)
(hpq : p < q) (A : ℝ) (a : Matrix (Fin p) (Fin q) ℤ)
(h_bound : ∀ k l, Complex.abs (a k l) ≤ A) :
∃ ξ : Fin q → ℤ, ξ ≠ 0
∧ ∀ k, (∑ l, a k l * ξ l = 0)
∧ ∀ l,
Complex.abs (ξ l) < c_1 * (1 + (c_1 * q * A ^ (p / (q - p))) ^ (1
    / (q - p))) :=
sorry
```

When prompted to remove commas by specifying all the `have` statements, the system encountered an issue where it consistently responded in the same manner. The system flagged the length of the message as excessive and advised shortening it for Mistral's handling. Subsequently, it became impossible to proceed with that particular chat session. Nevertheless, following a new chat session and several prompts, we succeeded in obtaining a reasonable proof sketch of the lemma.

```
variable {K : Type*} [Field K] [NumberField K] (b : Basis
    (Module.Free.ChooseBasisIndex ℤ (NumberField.ringOfIntegers
    K)) ℚ K)

instance : Algebra ℚ K := sorry

-- Define a theorem stating that the set of absolute values of the
--conjugates of an algebraic integer is nonempty
theorem nonempty_conjugates_abs (α : K) :
(Set.toFinset (Set.image Complex.abs (Polynomial.rootSet (minpoly
    ℚ α) ℂ))).Nonempty :=
sorry

-- Define the house of an algebraic integer
noncomputable def house (α : K) : ℝ :=
Finset.max' (Set.toFinset (Set.image Complex.abs
    (Polynomial.rootSet (minpoly ℚ α) ℂ)))
(nonempty_conjugates_abs α)

abbrev coeffs (α : K) : Fin (finrank ℚ K) → ℤ := sorry

-- Formalize the statement of the remark
theorem remark (α : K) :
∀ (i : Fin (finrank ℚ K)), |coeffs α i| ≤ c * house α := sorry

abbrev c₂ : ℤ := sorry

variable (b : Basis (Module.Free.ChooseBasisIndex ℤ
    (NumberField.ringOfIntegers K)) ℚ K)

variable (σ : K →+* ℂ)
```

```
theorem lemma82 (p q : ℕ) (hpq : 0 < p ∧ p < q) (A : ℝ) (hA : 1
    ≤ A)
(a : Matrix (Fin p) (Fin q) (𝒪 K)) (h_bound : ∀ k l, house
    ((algebraMap (𝒪 K) K) (a k l)) ≤ A) :
∃ ξ : Fin q → ℤ, ξ ≠ 0 ∧ ∀ k, (∑ l, a k l * ξ l = 0) ∧ ∀ l,
    Complex.abs (σ (ξ l)) < c₂ * (1 + (c₂ * q * A ^ (p / (h -
    p))) ^ (1 / (q - p))) :=
by
-- Define the matrix a′ by mapping each entry of a to its
    absolute value
have a′ : Matrix (Fin p) (Fin q) ℤ := sorry

-- Define the hypothesis ha′ by applying the house function to
    each entry of a
have ha′ : ∀ j k, |a′ j k| ≤ A := sorry

-- Apply Lemma 8.1 to get a non-trivial set of solutions to the
    system of equations
obtain ⟨x′, hx′, hx′_bound⟩ := lemma81 p q hpq A hA a′ ha′

-- Define ξ in terms of the coefficients x′
have ξ : Fin q → K := fun i => ∑ j, x′ i * b j

-- Expand the equation a * ξ = 0 and rearrange to get a system of
    hp equations with hq unknowns
have eq1 : ∀ k, ∑ l, a k l * ξ l = 0 := sorry

-- Bound the complex absolute values of the ξ′s using the
    triangle inequality and the bound on the x′s
have bound2 : ∀ l, Complex.abs (σ (ξ l)) < c₂ * (1 + (c₂ * q * A
    ^ (p / (h - p))) ^ (1 / (q - p))) := sorry

-- Set c₁ = c₂ and finish the proof
sorry
```