Leveraging Large Language Models for Autoformalizing Theorems: A Case Study

Michail Karatarakis

Radboud University Nijmegen, The Netherlands

Motivation

Large Language Models show promise in automating tasks in formal theorem proving, yet they encounter challenges such as inaccuracies and handling complex proofs. In our investigation, we employ Mistral for autoformalization in mathematics using the Lean 4 theorem prover [3] and its mathematical library, `mathlib` [1]. We focus on autoformalizing the following two theorems in number theory.

Theorem 01 (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 $a_{j1}x_1 + \cdots a_{jN}x_N = 0$ and $|x_k| \leq (NA)^{\frac{M}{N-M}}$.*

Theorem 02 (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 $\xi_1 \ldots, \xi_q$, not all zero, satisfying $a_{k1}\xi_1 + \cdots + a_{kq}\xi_q = 0$ and $\overline{|\xi_l|} < c_1(1 + (c_1 qA)^{\frac{p}{q-p}})$.*

Our aim is to generate formal proof sketches as the primary output. We first preprocess the source text [2] to extract relevant mathematical statements in LaTeX and utilize prompt engineering for autoformalization by initially excluding proofs.

We provide a centralized prompt repository[1], and the following list presents our observations from our experiments. For more details and the complete informal mathematical text and generated code, readers should consult Appendix 1.

Observations

- **Ensure precision and clarity** Ensuring proper text preprocessing for generating the correct definitions and proofs is equally significant as the act of prompting itself. One promising strategy for transcribing definitions involves modifying informal text to align with the formal definition we intend to use. It is always beneficial to review the existing content in `mathlib` to facilitate more informed decision-making in the preprocessing phase.
  For instance, in the proof of Lemma 01, $-B_j$ represents the sum of the negative coefficients of y_j. When prompted, Mistral autonomously generated the following definition:

```
let B : Fin M → ℤ
  | j => ∑ k, if a j k < 0 then -a j k else 0
```

  which is syntactically incorrect but undesirable in our case. After changing the informal text to " We define B_j as the sum of the $-min(0, a_{jk})$ for all a_{jk}.", Mistral changed its response to the following :

```
let B : Fin M → ℕ
  | j => ∑ k : Fin N, -min 0 (a j k)
```

  which is still syntactically incorrect but closer to what we want.
- **Adhere to Lean 4 syntax and conventions** Mistral's training data, constrained to Lean 3 syntax until January 2022, shapes its approach to syntax. Addressing this requires encompassing modifications to notation, replacement of outdated imports, and provision of examples demonstrating the correct syntax.
  For example, in the proof of Theorem 01, Mistral employed conventional mathematical notation, such as # for the cardinality of sets, which diverges from Lean's syntax.

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

```
-- Prove that the number of sets of x_k is (H + 1)^N
  have hXCount : # { x : Fin N → ℤ | ∀ k, 0 ≤ x k ∧ x k ≤ H } = (H + 1) ^ N :=
    sorry
```

In many cases, Mistral encountered difficulties in introducing certain notions, such as the aforementioned sum B_j, and many attempts led to the introduction of syntax that was invalid in both Lean 3 and Lean 4.

```
let B : Fin M → ℤ
B j := ∑ k, -min 0 (a j k)
```

To address such issues, we used examples as prompts.
`Here is an example of the correct syntax for "let" statements:`

```
let y : Fin M → ℤ
  | j => ∑ k : Fin N, a j k * x k
```

`should change to`

```
let y : Fin M → ℤ := fun j => ∑ k : Fin N, a j k * x k.
```

`Do the same for "C" and "B".`
Finally, after managing to bring the proof sketches of both theorems to a reasonable state, we were unable to ensure that Mistral adhered faithfully to Lean 4 syntax despite our continuous efforts. In some cases, Mistral would refuse to replace the `begin` with `by` and to eliminate the `end` commands. In other cases, Mistral would agree to make these modifications but refuse to remove the commas at the end of the sub-proof statements.

– **Handling Type Mismatch Errors** Type mismatch errors, commonly encountered in theorem proving, indicate disparities between expected and actual types. For instance, Mistral might define B_j as:

```
def B : Fin M → ℕ := fun j => ∑ k : Fin N, -min 0 (a j k)
```

and the type of `B` had to be updated to `Fin M → ℤ` to match the type of `a`.

– **Addressing Autonomy Challenges** As we have already seen, one of the main challenges lies in LLMs autonomously generating the prerequisite definitions and theorems.

1. **Definitions:** Establishing definitions is widely recognized as one of the most challenging aspects of formalization. For instance, Theorem 01 concerns rational integers, but a single attempt resulted in the following definition:

   ```
   abbrev RationalInteger := Int
   ```

   despite Mistral's understanding that a rational integer is simply an integer.
   The issues that are most difficult to solve arise when the model is prompted to formalize definitions that it hasn't encountered before. For example, let K be an algebraic number field of degree h, and let α be an algebraic integer in K. We shall denote by $\overline{|\alpha|}$ the maximum of the modulus of the conjugates $\alpha^{(i)}$ with $1 \leq i \leq h$ of α, that is, $\overline{|\alpha|} = \max_{1 \leq i \leq h} \left|\alpha^{(i)}\right|$.
   We aim to assist Mistral in formulating the definition of $\overline{|\alpha|}$. Following the initial prompt, the expected response results to a familiar pattern, with Mistral attempting independently and selectively to delineate the prerequisite definitions.

   ```
   -- 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'
   ```

   The customary approach to tackling these challenges involves supplying the LLM with definitions sourced from existing libraries such as `mathlib`, or ones provided by the user.
2. **Proofs** Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously. Common issues include selectively proving sub-proofs, using arbitrary identifiers, mixing Lean 3 and Lean 4 syntax, and utilizing theorems from Lean's 3 `mathlib`. Thus, it's important to review the informal proof steps, break down complex informal proofs into manageable components, and supply the LLM with the relevant theorems from `mathlib`, or ones provided by the user.

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)

1 Appendix

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

Theorem 11 (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 several prompts, the system successfully produced the correct syntax and came up with the statement and a proof sketch of the lemma. Through further iterations, we achieved a stage where the autoformalization of both the primary statement and its sub-proof statements became reasonably accurate.

```
theorem lemma81 (M N : ℕ) (hMN : 0 < M ∧ M < N) (A : ℝ) (hA : 1 ≤ A)
  (a : Matrix (Fin M) (Fin N) ℤ) (ha : ∀ j k, |a j k| ≤ A) :
  ∃ x : Fin N → ℤ, (∃ k, x k ≠ 0) ∧ ∀ j, ∑ k, a j k * x k = 0
  ∧ ∀ k, |x k| ≤ (N * A)^(M / (N - M)) := by
  -- Define H
  let H := (N * A) ^ (M / (N - M))

  -- Define X as the set of all sets of x_k satisfying 0 ≤ x_k ≤ H
  let X : Set (Fin N → ℤ) := { x | ∀ k, 0 ≤ x k ∧ x k ≤ H }

```

```
-- Define Y as the set of all corresponding sets of y_j
let Y : Set (Fin M → ℤ) := { y | ∃ x ∈ X, ∀ j, y j = ∑ k, a j k * x k }

-- Prove that NA < (H+1)^((N-M)/M)
have hNA : N * A < (H + 1) ^ ((N - M) / M) := by sorry

-- Prove that (NAH) + 1 ≤ NA(H + 1)
have hNAH : (N * A * H) + 1 ≤ N * A * (H + 1) := by sorry

-- Prove that NA(H + 1) < (H + 1)^(N/M)
have hNAH' : N * A * (H + 1) < (H + 1) ^ (N / M) := by sorry

-- Define y_j for a given x
let y (x : Fin N → ℤ) : Fin M → ℤ := fun j => ∑ k : Fin N, a j k * x k

-- Define B_j and C_j
let B : Fin M → ℤ := fun j => ∑ k : Fin N, -min 0 (a j k)

let C : Fin M → ℤ := fun j => ∑ k : Fin N, max 0 (a j k)

-- 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 x j ∧ y x j ≤ C j * H ∧ B j + C j ≤ N * A := by sorry

-- Prove that X and Y have finite cardinalities
haveI : Fintype X := by sorry

haveI : Fintype Y := by sorry

-- Prove that the number of sets of x_k is (H + 1)^N
have hXCount : Finset.card X.toFinset = (H + 1) ^ N := by sorry

-- Prove that the number of corresponding sets of y_j is at most (NAH + 1)^M
have hYCount : Finset.card Y.toFinset ≤ (N * A * H + 1) ^ M := by sorry

-- Prove that the cardinality of X is larger than the cardinality of Y
have hCardXY : Finset.card X.toFinset > Finset.card Y.toFinset := by sorry

-- Apply the pigeonhole principle to find two sets of x_k that correspond to the
  same set of y_j
have hPigeonhole : ∃ (x' x'' : Fin N → ℤ) (_ : x' ∈ X) (_ : x'' ∈ X), ∀ j, y x' j = y
  x'' j := by sorry

-- Extract the two sets of x_k that correspond to the same set of y_j
rcases hPigeonhole with ⟨x', x'', hx', hx''⟩

-- 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 : Fin M), ∑ k : Fin N, a j k * x k = 0
 ∧ ∀ (k : Fin N), |x k| ≤ (↑N * A) ^ (M / (N - M)) := sorry

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

Example 2

Let K be an algebraic number field of degree h, and let $\beta_1, \ldots, \beta_h$ be an integer basis, so that every integer in K has the unique representation $a_1\beta_1 + \ldots + a_h\beta_h$ where $a_1, \ldots, a_h$ are rational integers. We

shall denote by $\overline{|\alpha|}$ the maximum of the modulus of the conjugates $\alpha^{(i)}$ with $(1 \leq i \leq h)$ of α, that is

$$\overline{|\alpha|} = \max_{1 \leq i \leq h} \left|\alpha^{(i)}\right|.$$

The subsequent step 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 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 the following prompts were formulated accordingly. As a result, 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)
```

The next theorem we aim to address is as follows:

Theorem 12 (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 $\xi_1 \ldots, \xi_q$, 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

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

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

$$0 = \sum_{l=1}^{q} a_{kl}\xi_l \tag{10}$$

$$= \sum_{n=1}^{q} a_{kl} \sum_{r=1}^{h} x_{lr}\beta_r \tag{11}$$

$$= \sum_{n=1}^{h} \sum_{l=1}^{q} x_{lr} \sum_{r=1}^{h} a_{aklru}\beta_u \tag{12}$$

$$= \sum_{u=1}^{h} \left(\sum_{r=1}^{h} \sum_{l=1}^{q} a_{klru} x_{lr} \right) \beta_u \tag{13}$$

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

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

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

$$\overline{|\, a_{klru} \,|} \le c \max_{1 \le i \le h} \beta^{(i)} A \tag{15}$$

$$\le c_2 A. \tag{16}$$

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

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

Therefore

$$\overline{|\, \xi_l \,|} \le |\, x_{l1} \,| \, \overline{|\, \beta_1 \,|} + \cdots + |\, x_{lh} \,| \, \overline{|\, \beta_h \,|}$$

$$\le c_2 h (1 + (hqc_2A)^{\frac{p}{(q-p)}}). \tag{18}$$

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

After some prompting we managed to get the statement of the theorem almost correct but when prompted to remove commas by specifying all the `have` statements, the system consistently responded in the same manner – the system flagged the length of the message as excessive and advised shortening it for handling. Consequently, 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
```

Despite our continuous efforts to prompt, we were unable to ensure that the LLM adhered to Lean 4 syntax. In some cases Mistral would refuse to replace the `begin` with `by` and to eliminate the `end` commands. In other cases, it would agree to make this modification but refuse to remove the commas at the end of the sub-proof statements.

2 Conclusion

During our investigation, establishing necessary concepts and prerequisite theorems posed challenges, requiring guidance and adjustments to conform to established conventions within the `Lean 4` and `mathlib` frameworks. To streamline the autoformalization of mathematics in Lean 4, we suggest the following strategies:

1. Provide clear and unambiguous mathematical statements and employ standard mathematical notation to facilitate better understanding and accurate Lean 4 code generation.
2. Utilize Lean 4 syntax and conventions to ensure adherence to best practices and enhance code readability.

3. Include relevant mathematical definitions and theorems theorems to provide context and improve the accuracy of Lean 4 code.
4. Present examples of the mathematical concepts to be formalized to facilitate understanding and improve code accuracy.
5. Simplify complex proofs into smaller, manageable components to enhance comprehension and promote accurate code generation.
6. Evaluate the generated Lean 4 code, provide feedback, and iterate as necessary to enhance comprehension and accuracy in subsequent iterations.