Leveraging Large Language Models for Autoformalizing Theorems: A Case Study.

9th Conference on Artificial Intelligence and Theorem Proving
September 1-6, 2024, Aussois, France

Michail Karatarakis,
Radboud University Nijmegen, The Netherlands

September 5, 2024

Project idea

I am interested in formalizing mathematics (number theory) in Lean, and I am not currently involved in AI — but I am trying to get into it.

Project idea

I am interested in formalizing mathematics (number theory) in Lean, and I am not currently involved in AI — but I am trying to get into it.

I visited Cezary at Innsbruck (April 2024) and met with Josef and Chad.

Project idea

I am interested in formalizing mathematics (number theory) in Lean, and I am not currently involved in AI — but I am trying to get into it.

I visited Cezary at Innsbruck (April 2024) and met with Josef and Chad.

Josef suggested the following project:

Project idea

I am interested in formalizing mathematics (number theory) in Lean, and I am not currently involved in AI — but I am trying to get into it.

I visited Cezary at Innsbruck (April 2024) and met with Josef and Chad.

Josef suggested the following project:

Do a case study by attempting to autoformalize these two lemmata in Lean 4 using Mistral.

Project idea

I am interested in formalizing mathematics (number theory) in Lean, and I am not currently involved in AI — but I am trying to get into it.

I visited Cezary at Innsbruck (April 2024) and met with Josef and Chad.

Josef suggested the following project:

Do a case study by attempting to autoformalize these two lemmata in Lean 4 using Mistral.

Aim : Try out an LLM (Mistral) to gain insights into how to improve it.

Methodology

- Extracted the mathematical statements from the source text (e.g., book [1, pp. 489-490], paper, etc.) in LaTeX.

Methodology

- Extracted the mathematical statements from the source text (e.g., book [1, pp. 489-490], paper, etc.) in LaTeX.
- We didn't train any model ourselves – just used Mistral.

Methodology

- Extracted the mathematical statements from the source text (e.g., book [1, pp. 489-490], paper, etc.) in LaTeX.
- We didn't train any model ourselves – just used Mistral.
- There was no fine-tuning.

Methodology

- Extracted the mathematical statements from the source text (e.g., book [1, pp. 489-490], paper, etc.) in LaTeX.
- We didn't train any model ourselves – just used Mistral.
- There was no fine-tuning.
- Applied prompt engineering to generate a formal proof sketch (not the actual proofs).

Methodology

- Extracted the mathematical statements from the source text (e.g., book [1, pp. 489-490], paper, etc.) in LaTeX.
- We didn't train any model ourselves – just used Mistral.
- There was no fine-tuning.
- Applied prompt engineering to generate a formal proof sketch (not the actual proofs).
- Stored prompts in a repository for future analysis.

Methodology

- Extracted the mathematical statements from the source text (e.g., book [1, pp. 489-490], paper, etc.) in LaTeX.
- We didn't train any model ourselves – just used Mistral.
- There was no fine-tuning.
- Applied prompt engineering to generate a formal proof sketch (not the actual proofs).
- Stored prompts in a repository for future analysis.

Lemma 1 (Siegel's Lemma)

There exists a "small" integral non-zero solution of a non-trivial underdetermined system of linear equations with integer coefficients.

Theorem

Let $0 < M < N$*, and* a_{jk} *be rational integers satisfying* $\left|a_{jk}\right| \leq A \quad$ *where* $1 \leq A$, $1 \leq j \leq M$ *and* $1 \leq k \leq N$*. Then there exists a set of rational integers* $x_1 \ldots, x_N$*, not all zero, satisfying* $a_{j1}x_1 + \cdots a_{jN}x_N = 0$ *and*

$|x_k| \leq (NA)^{\frac{M}{N-M}}$.

Lemma 1 (Siegel's Lemma)

There exists a "small" integral non-zero solution of a non-trivial underdetermined system of linear equations with integer coefficients.

Theorem

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* $x_1 \ldots, x_N$, *not all zero, satisfying* $a_{j1}x_1 + \cdots a_{jN}x_N = 0$ *and*

$|x_k| \leq (NA)^{\frac{M}{N-M}}$.

Non-mathlib version:

```
theorem siegel.{u_2, u_1} {α : Type u_1} {β : Type u_2} [Fintype α] [Fintype β]
[DecidableEq β] [DecidableEq α] {M N : ℕ} (cardα : Fintype.card α = M)
(cardβ : Fintype.card β = N)
(hOM : 0 < M) (hMN : M < N) (a : Matrix α β ℤ) (A : ℝ) (hA : 1 ≤ A)
(habs : ∀ (j : α) (k : β), ↑|a j k| ≤ A) :
∃ x, (∃ k, x k ≠ 0) ∧ Matrix.mulVec a x = 0 ∧
∀ (k : β), ↑|x k| ≤ (↑N * A) ^ (↑M / (↑N - ↑M))
```

Lemma 1 (Siegel's Lemma)

There exists a "small" integral non-zero solution of a non-trivial underdetermined system of linear equations with integer coefficients.

Theorem

Let $0 < M < N$, *and* a_{jk} *be rational integers satisfying* $\left|a_{jk}\right| \leq A$ *where* $1 \leq A$, $1 \leq j \leq M$ *and* $1 \leq k \leq N$. *Then there exists a set of rational integers* $x_1 \ldots, x_N$, *not all zero, satisfying* $a_{j1}x_1 + \cdots a_{jN}x_N = 0$ *and*

$|x_k| \leq (NA)^{\frac{M}{N-M}}$.

Non-mathlib version:

```
theorem siegel.{u_2, u_1} {α : Type u_1} {β : Type u_2} [Fintype α] [Fintype β]
[DecidableEq β] [DecidableEq α] {M N : ℕ} (cardα : Fintype.card α = M)
(cardβ : Fintype.card β = N)
(h0M : 0 < M) (hMN : M < N) (a : Matrix α β ℤ) (A : ℝ) (hA : 1 ≤ A)
(habs : ∀ (j : α) (k : β), ↑|a j k| ≤ A) :
∃ x, (∃ k, x k ≠ 0) ∧ Matrix.mulVec a x = 0 ∧
∀ (k : β), ↑|x k| ≤ (↑N * A) ^ (↑M / (↑N - ↑M))
```

Mistral's final version:

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

Lemma 1 (Siegel's Lemma)

There exists a "small" integral non-zero solution of a non-trivial underdetermined system of linear equations with integer coefficients.

Theorem

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* $x_1 \ldots, x_N$, *not all zero, satisfying* $a_{j1}x_1 + \cdots a_{jN}x_N = 0$ *and*

$|x_k| \leq (NA)^{\frac{M}{N-M}}$.

Non-mathlib version:

```
theorem siegel.{u_2, u_1} {α : Type u_1} {β : Type u_2} [Fintype α] [Fintype β]
[DecidableEq β] [DecidableEq α] {M N : ℕ} (cardα : Fintype.card α = M)
(cardβ : Fintype.card β = N)
(hOM : 0 < M) (hMN : M < N) (a : Matrix α β ℤ) (A : ℝ) (hA : 1 ≤ A)
(habs : ∀ (j : α) (k : β), ↑|a j k| ≤ A) :
∃ x, (∃ k, x k ≠ 0) ∧ Matrix.mulVec a x = 0 ∧
∀ (k : β), ↑|x k| ≤ (↑N * A) ^ (↑M / (↑N - ↑M))
```

Mistral's final version:

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

Mathlib version:

```
theorem Int.Matrix.exists_ne_zero_int_vec_norm_le.{u_1, u_2} {α : Type u_1}
    {β : Type u_2} [Fintype α] [Fintype β] [DecidableEq β] [DecidableEq α]
    (A : Matrix α β ℤ) (hn : Fintype.card α < Fintype.card β)
    (hm : 0 < Fintype.card α) :
  ∃ t, t ≠ 0 ∧ A.mulVec t = 0 ∧ ‖t‖ ≤ (↑(Fintype.card β) * max 1 ‖A‖)
  ^ (↑(Fintype.card α) / (↑(Fintype.card β) - ↑(Fintype.card α))) := ...
```

Lemma 2

Theorem

Let $0 < p < q$, and a_{kl} be integers in K 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 algebraic integers $\xi_1 \ldots, \xi_q$, not all zero, satisfying $a_{k1}\xi_1 + \cdots + a_{kq}\xi_q = 0, \quad 1 \leq k \leq p, \quad 1 \leq l \leq q$ and

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

Lemma 2

Theorem

Let $0 < p < q$, and a_{kl} be integers in K 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 algebraic integers $\xi_1 \ldots, \xi_q$, not all zero, satisfying $a_{k1}\xi_1 + \cdots + a_{kq}\xi_q = 0, \quad 1 \leq k \leq p, \quad 1 \leq l \leq q$ and

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

Mistral's final version:

```
theorem lemma82 (p q : ℕ) (hpq : 0 < p ∧ p < q) (A : ℝ) (hA : 1 ≤ A)
(a : Matrix (Fin p) (Fin q) (𝒪 K)) (σ : K →+* ℂ)
(h_bound : ∀ k l, house ((algebraMap (𝒪 K) K) (a k l)) ≤ A) :
∃ ξ : Fin q → ℤ, ξ ≠ 0 ∧ ∀ k, (Σ l, a k l * ξ l = 0) ∧
∀ l, Complex.abs (σ (ξ l)) <
   c₂ * (1 + (c₂ * q * A ^ (p / (h - p))) ^ (1 / (q - p))) := sorry
```

Lemma 2

Theorem

Let $0 < p < q$, and a_{kl} be integers in K 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 algebraic integers $\xi_1 \ldots, \xi_q$, not all zero, satisfying $a_{k1}\xi_1 + \cdots + a_{kq}\xi_q = 0, \quad 1 \leq k \leq p, \quad 1 \leq l \leq q$ and

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

Mistral's final version:

```
theorem lemma82 (p q : ℕ) (hpq : 0 < p ∧ p < q) (A : ℝ) (hA : 1 ≤ A)
(a : Matrix (Fin p) (Fin q) (𝒪 K)) (σ : K →+* ℂ)
(h_bound : ∀ k l, house ((algebraMap (𝒪 K) K) (a k l)) ≤ A) :
∃ ξ : Fin q → ℤ, ξ ≠ 0 ∧ ∀ k, (Σ l, a k l * ξ l = 0) ∧
∀ l, Complex.abs (σ (ξ l)) <
   c₂ * (1 + (c₂ * q * A ^ (p / (h - p))) ^ (1 / (q - p))) := sorry
```

Mathlib version:

```
theorem NumberField.exists_ne_zero_int_vec_house_le.{u_1, u_2, u_3}
  {p q : ℕ} {A : ℝ} (K : Type u_1) [Field K] [NumberField K]
  [DecidableEq (K →+* ℂ)] {α : Type u_3} {β : Type u_2} [Fintype α]
  [Fintype β] (a : Matrix α β (𝒪 K)) (cardα : Fintype.card α = p)
  (cardβ : Fintype.card β = q) (h0p : 0 < p) (hpq : p < q) :
∃ ξ, ξ ≠ 0 ∧ Matrix.mulVec a  ξ = 0 ∧
∀ (l : β), house ↑(ξ l) ≤ NumberField.c₁ K * (NumberField.c₁ K * ↑q * A
     ) ^ (↑p / (↑q - ↑p)) := ...
```

Addressing Autonomy Challenges - Definitions

Establishing definitions is widely recognized as one of the most challenging aspects of (auto)formalization.

Addressing Autonomy Challenges - Definitions

Establishing definitions is widely recognized as one of the most challenging aspects of (auto)formalization.

In mathematics, an object can be defined in many different ways.

Addressing Autonomy Challenges - Definitions

Establishing definitions is widely recognized as one of the most challenging aspects of (auto)formalization.

In mathematics, an object can be defined in many different ways.

It's difficult to handle the bijections between different definitions.

Addressing Autonomy Challenges - Definitions

Establishing definitions is widely recognized as one of the most challenging aspects of (auto)formalization.

In mathematics, an object can be defined in many different ways.

It's difficult to handle the bijections between different definitions.

The most challenging issues arise when the model is prompted to formalize definitions that it has not encountered before.

Addressing Autonomy Challenges - Definitions

Establishing definitions is widely recognized as one of the most challenging aspects of (auto)formalization.

In mathematics, an object can be defined in many different ways.

It's difficult to handle the bijections between different definitions.

The most challenging issues arise when the model is prompted to formalize definitions that it has not encountered before.

After the initial prompt, the expected response follows a familiar pattern, with Mistral attempting independently and selectively to delineate the prerequisite definitions.

House of an algebraic integer

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

House of an algebraic integer

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

- Non-mathlib version

```
theorem rootSet_abs_nonempty (α : K) :
    (toFinset (Complex.abs '' rootSet (minpoly ℚ α) ℂ)).Nonempty := by
  rw [toFinset_nonempty]
  apply Set.Nonempty.image
  rw ←[ range_eval_eq_rootSet_minpoly]
  apply range_nonempty

/-- The house of ‘α‘ is the largest modulus of its conjugates.-/
abbrev House (α : K) : ℝ :=
  (Complex.abs '' rootSet (minpoly ℚ α) ℂ).toFinset.max' (rootSet_abs_nonempty α)
```

House of an algebraic integer

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

- Non-mathlib version

```
theorem rootSet_abs_nonempty (α : K) :
    (toFinset (Complex.abs '' rootSet (minpoly ℚ α) ℂ)).Nonempty := by
  rw [toFinset_nonempty]
  apply Set.Nonempty.image
  rw ←[ range_eval_eq_rootSet_minpoly]
  apply range_nonempty

/-- The house of `α` is the largest modulus of its conjugates.-/
abbrev House (α : K) : ℝ :=
  (Complex.abs '' rootSet (minpoly ℚ α) ℂ).toFinset.max' (rootSet_abs_nonempty α)
```

- Mistra's first version

```
-- Define the conjugates of an algebraic integer
def conjugates {K : Type u_1} [Field K] [NumberField K] (α : K) : List K :=
  -- insert definition here
-- Define the house of an algebraic integer
def house {K : Type u_1} [Field K] [NumberField K] (α : K) : ℝ :=
  (conjugates α).map (fun x => abs x).max'
```

House of an algebraic integer

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

- Non-mathlib version

```
theorem rootSet_abs_nonempty (α : K) :
    (toFinset (Complex.abs '' rootSet (minpoly ℚ α) ℂ)).Nonempty := by
  rw [toFinset_nonempty]
  apply Set.Nonempty.image
  rw ←[ range_eval_eq_rootSet_minpoly]
  apply range_nonempty

/-- The house of `α` is the largest modulus of its conjugates.-/
abbrev House (α : K) : ℝ :=
  (Complex.abs '' rootSet (minpoly ℚ α) ℂ).toFinset.max' (rootSet_abs_nonempty α)
```

- Mistra's first version

```
-- Define the conjugates of an algebraic integer
def conjugates {K : Type u_1} [Field K] [NumberField K] (α : K) : List K :=
  -- insert definition here
-- Define the house of an algebraic integer
def house {K : Type u_1} [Field K] [NumberField K] (α : K) : ℝ :=
  (conjugates α).map (fun x => abs x).max'
```

- Mistral's final version

```
-- Define the house of an algebraic integer
def house {K : Type u_1} [Field K] [NumberField K] (α : K) : ℝ :=
  let embeddings := NumberField.Embeddings.range_eval_eq_rootSet_minpoly α
  (embeddings.map (fun x => abs x)).max'
```

House of an algebraic integer

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 $(1 \le i \le h)$ of α, that is $\overline{|\alpha|} = \max_{1\le i\le h} \left|\alpha^{(i)}\right|$.

- Non-mathlib version

```
theorem rootSet_abs_nonempty (α : K) :
    (toFinset (Complex.abs '' rootSet (minpoly ℚ α) ℂ)).Nonempty := by
  rw [toFinset_nonempty]
  apply Set.Nonempty.image
  rw ←[ range_eval_eq_rootSet_minpoly]
  apply range_nonempty

/-- The house of ‘α‘ is the largest modulus of its conjugates.-/
abbrev House (α : K) : ℝ :=
  (Complex.abs '' rootSet (minpoly ℚ α) ℂ).toFinset.max' (rootSet_abs_nonempty α)
```

- Mistra's first version

```
-- Define the conjugates of an algebraic integer
def conjugates {K : Type u_1} [Field K] [NumberField K] (α : K) : List K :=
  -- insert definition here
-- Define the house of an algebraic integer
def house {K : Type u_1} [Field K] [NumberField K] (α : K) : ℝ :=
  (conjugates α).map (fun x => abs x).max'
```

- Mistral's final version

```
-- Define the house of an algebraic integer
def house {K : Type u_1} [Field K] [NumberField K] (α : K) : ℝ :=
  let embeddings := NumberField.Embeddings.range_eval_eq_rootSet_minpoly α
  (embeddings.map (fun x => abs x)).max'
```

- mathlib version

```
/-- The house of an algebraic number as the norm of its image by the canonical embedding.
    -/
def house {K : Type u_1} [Field K] [NumberField K] (α : K) : ℝ :=
  ‖canonicalEmbedding K α‖
```

Ensure Precision and Clarity

- Ensuring proper text preprocessing for generating correct definitions and proofs is just as important as the act of prompting itself.

Ensure Precision and Clarity

- Ensuring proper text preprocessing for generating correct definitions and proofs is just as important as the act of prompting itself.
- Be prepared to address syntax issues. Mistral's training data, likely constrained largely to Lean 3 syntax (until January 2022), shaped its approach to syntax. We had to change the notation, replace outdated imports, and provide examples demonstrating the correct syntax.

Ensure Precision and Clarity

- Ensuring proper text preprocessing for generating correct definitions and proofs is just as important as the act of prompting itself.
- Be prepared to address syntax issues. Mistral's training data, likely constrained largely to Lean 3 syntax (until January 2022), shaped its approach to syntax. We had to change the notation, replace outdated imports, and provide examples demonstrating the correct syntax.
- Review the existing content in `mathlib` to make more informed decisions.

Ensure Precision and Clarity

- Ensuring proper text preprocessing for generating correct definitions and proofs is just as important as the act of prompting itself.
- Be prepared to address syntax issues. Mistral's training data, likely constrained largely to Lean 3 syntax (until January 2022), shaped its approach to syntax. We had to change the notation, replace outdated imports, and provide examples demonstrating the correct syntax.
- Review the existing content in `mathlib` to make more informed decisions.
- Ask on Zulip what the best approach would be.

Ensure Precision and Clarity

- Ensuring proper text preprocessing for generating correct definitions and proofs is just as important as the act of prompting itself.
- Be prepared to address syntax issues. Mistral's training data, likely constrained largely to Lean 3 syntax (until January 2022), shaped its approach to syntax. We had to change the notation, replace outdated imports, and provide examples demonstrating the correct syntax.
- Review the existing content in `mathlib` to make more informed decisions.
- Ask on Zulip what the best approach would be.
- Provide the LLM with definitions sourced from existing libraries such as `mathlib`, or define them ourselves.

Ensure Precision and Clarity

- Ensuring proper text preprocessing for generating correct definitions and proofs is just as important as the act of prompting itself.
- Be prepared to address syntax issues. Mistral's training data, likely constrained largely to Lean 3 syntax (until January 2022), shaped its approach to syntax. We had to change the notation, replace outdated imports, and provide examples demonstrating the correct syntax.
- Review the existing content in `mathlib` to make more informed decisions.
- Ask on Zulip what the best approach would be.
- Provide the LLM with definitions sourced from existing libraries such as `mathlib`, or define them ourselves.
- Modify informal text to align with the definitions we intend to use.

Ensure precision and clarity - Example

For instance, in the proof of Lemma 1, $-B_j$ represents the sum of the negative coefficients of y_j. When prompted, Mistral autonomously generated the following definition:

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

which is syntactically correct but undesirable in our case.

Ensure precision and clarity - Example

For instance, in the proof of Lemma 1, $-B_j$ represents the sum of the negative coefficients of y_j. When prompted, Mistral autonomously generated the following definition:

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

which is syntactically correct 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 => Σ k : Fin N, -min 0 (a j k)
```

which is syntactically correct but the type is wrong since a_{jk} is an integer even if Mistral used the correct type in the previous attempt.

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

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

Proof sketch of Lemma 1

```
theorem lemma81 (M N : ) (hMN : 0 < M ∧ M < N) (A : ) (hA : 1 ≤ A)
  (a : Matrix (Fin M) (Fin N) ) (ha : ∀ j k, |a j k| ≤ A) :
  ∃ x : Fin N → , (∃ k, x k ≠ 0) ∧ ∀ j, Σ 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 → ) := { 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 → ) := { y | ∃ x ∈ X, ∀ j, y j = Σ 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 → ) : Fin M →  := fun j => Σ k : Fin N, a j k * x k
-- Define B_j and C_j
  let B : Fin M →  := fun j => Σ k : Fin N, -min 0 (a j k)
  let C : Fin M →  := fun j => Σ 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 → ), (∀ 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 → ) (_ : 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 →  := fun k => x' k - x'' k
-- Prove that x_k satisfies the required conditions
  have hX : (∃ k, x k ≠ 0) ∧ ∀ (j : Fin M), Σ 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⟩
```

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

  In Lemma 2, Mistral agreed to make these modifications but refused to remove the commas at the end of the sub-proof statements.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

  In Lemma 2, Mistral agreed to make these modifications but refused to remove the commas at the end of the sub-proof statements.
- Selectively proving sub-proofs.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

  In Lemma 2, Mistral agreed to make these modifications but refused to remove the commas at the end of the sub-proof statements.
- Selectively proving sub-proofs.
- Using arbitrary identifiers (that are not in `mathlib`).

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

  In Lemma 2, Mistral agreed to make these modifications but refused to remove the commas at the end of the sub-proof statements.
- Selectively proving sub-proofs.
- Using arbitrary identifiers (that are not in `mathlib`).
- Mixing Lean 3 and Lean 4 syntax.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

  In Lemma 2, Mistral agreed to make these modifications but refused to remove the commas at the end of the sub-proof statements.
- Selectively proving sub-proofs.
- Using arbitrary identifiers (that are not in `mathlib`).
- Mixing Lean 3 and Lean 4 syntax.
- Utilizing theorems from Lean 3's `mathlib`.

Proofs

Dealing with proofs poses challenges due to Lean's syntax and the LLM's tendency to prove things autonomously.

Issues include:

- We were unable to ensure that Mistral adhered strictly to Lean 4 syntax despite continuous efforts.

  In Lemma 1, Mistral refused to replace `begin` with `by` and did not eliminate the `end` commands.

  In Lemma 2, Mistral agreed to make these modifications but refused to remove the commas at the end of the sub-proof statements.
- Selectively proving sub-proofs.
- Using arbitrary identifiers (that are not in `mathlib`).
- Mixing Lean 3 and Lean 4 syntax.
- Utilizing theorems from Lean 3's `mathlib`.
- Laziness – many proofs were of the form:

```
theorem foo : bar := by
-- TODO: continue proof
sorry
```

Dealing with Proofs

- Review the informal proof steps to see if they can be simplified or improved.

Dealing with Proofs

- Review the informal proof steps to see if they can be simplified or improved.
- Break down complex informal proofs into manageable components (fill in gaps in your source text, preprocess the text, etc.).

Dealing with Proofs

- Review the informal proof steps to see if they can be simplified or improved.
- Break down complex informal proofs into manageable components (fill in gaps in your source text, preprocess the text, etc.).
- Provide the LLM with the relevant theorems from `mathlib`, or use ones provided by the user.

Dealing with Proofs

- Review the informal proof steps to see if they can be simplified or improved.
- Break down complex informal proofs into manageable components (fill in gaps in your source text, preprocess the text, etc.).
- Provide the LLM with the relevant theorems from `mathlib`, or use ones provided by the user.
- Ensuring that the LLM is trained on the proof assistant or the version of the proof assistant you intend to use is beneficial.

The Formalizations of the Two Lemmata

What was required to complete the formalization of these lemmata was:

The Formalizations of the Two Lemmata

What was required to complete the formalization of these lemmata was:

- Lemma 1 (`mathlib` version, ~ 200 LOC): Barroero, Capuano, Turchet, ..., maintainers (Brasca, ...) etc.

The Formalizations of the Two Lemmata

What was required to complete the formalization of these lemmata was:

- Lemma 1 (`mathlib` version, ~ 200 LOC): Barroero, Capuano, Turchet, ..., maintainers (Brasca, ...) etc.
- Lemma 2 (`mathlib` version, ~ 300 LOC): Karatarakis, `mathlib` reviewers Brasca, Barroero, Roblot, ..., and Mario's "voodoo black magic".

The Formalizations of the Two Lemmata

What was required to complete the formalization of these lemmata was:

- Lemma 1 (`mathlib` version, $\sim$ 200 LOC): Barroero, Capuano, Turchet, ..., maintainers (Brasca, ...) etc.
- Lemma 2 (`mathlib` version, $\sim$ 300 LOC): Karatarakis, `mathlib` reviewers Brasca, Barroero, Roblot, ..., and Mario's "voodoo black magic".

For autoformalization, what we want in the future is the following:

The Formalizations of the Two Lemmata

What was required to complete the formalization of these lemmata was:

- Lemma 1 (`mathlib` version, $\sim$ 200 LOC): Barroero, Capuano, Turchet, ..., maintainers (Brasca, ...) etc.
- Lemma 2 (`mathlib` version, $\sim$ 300 LOC): Karatarakis, `mathlib` reviewers Brasca, Barroero, Roblot, ..., and Mario's "voodoo black magic".

For autoformalization, what we want in the future is the following:

- Lemma 1: user(s), AI tool.

The Formalizations of the Two Lemmata

What was required to complete the formalization of these lemmata was:

- Lemma 1 (`mathlib` version, $\sim$ 200 LOC): Barroero, Capuano, Turchet, ..., maintainers (Brasca, ...) etc.
- Lemma 2 (`mathlib` version, $\sim$ 300 LOC): Karatarakis, `mathlib` reviewers Brasca, Barroero, Roblot, ..., and Mario's "voodoo black magic".

For autoformalization, what we want in the future is the following:

- Lemma 1: user(s), AI tool.
- Lemma 2: user(s), AI tool.

Observations

Attempting to formalize a theorem that hasn't been formalized before can be challenging.

Observations

Attempting to formalize a theorem that hasn't been formalized before can be challenging.

Attempting to autoformalize a theorem that hasn't been formalized before may currently be out of reach.

Observations

Attempting to formalize a theorem that hasn't been formalized before can be challenging.

Attempting to autoformalize a theorem that hasn't been formalized before may currently be out of reach.

Attempting to autoformalize a theorem that has already been formalized is more promising (because we can "cheat").

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?
- Is there a unified library we can train on? (`mathlib`, ...)

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?
- Is there a unified library we can train on? (`mathlib`, ...)
- Are the prerequisite definitions available in the library?

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?
- Is there a unified library we can train on? (`mathlib`, ...)
- Are the prerequisite definitions available in the library?
- For the proofs, is there automation? (hammers, metaprogramming ...)

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?
- Is there a unified library we can train on? (`mathlib`, ...)
- Are the prerequisite definitions available in the library?
- For the proofs, is there automation? (hammers, metaprogramming ...)
- Is there a way to seek guidance from system experts, AI engineers, or other mathematicians? (Zulip, ...)

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?
- Is there a unified library we can train on? (`mathlib`, ...)
- Are the prerequisite definitions available in the library?
- For the proofs, is there automation? (hammers, metaprogramming ...)
- Is there a way to seek guidance from system experts, AI engineers, or other mathematicians? (Zulip, ...)
- Can we develop proof-assistant specific AI tools (e.g., implement ML algorithms, prove their correctness)?

Conclusions / Food for Thought

- We suggest that maths autoformalization should be proof-assistant specific.
- Did the model train on the version of the proof assistant or the proof assistant we are attempting to use?
- Is there a unified library we can train on? (`mathlib`, ...)
- Are the prerequisite definitions available in the library?
- For the proofs, is there automation? (hammers, metaprogramming ...)
- Is there a way to seek guidance from system experts, AI engineers, or other mathematicians? (Zulip, ...)
- Can we develop proof-assistant specific AI tools (e.g., implement ML algorithms, prove their correctness)?
- It is important to focus on definitions and prerequisites.

Evangelism for (Auto)formalization

Let's assume that mathematics (auto)formalization is part of the future.

Evangelism for (Auto)formalization

Let's assume that mathematics (auto)formalization is part of the future.

Some mathematicians feel that interactive theorem provers (ITPs) are not worth the effort.

Evangelism for (Auto)formalization

Let's assume that mathematics (auto)formalization is part of the future.

Some mathematicians feel that interactive theorem provers (ITPs) are not worth the effort.

There is a perception among many mathematicians that current AI tools are far from proving their theorems, or that the theorems being proven by AI are not of particular interest.

Evangelism for (Auto)formalization

Let's assume that mathematics (auto)formalization is part of the future.

Some mathematicians feel that interactive theorem provers (ITPs) are not worth the effort.

There is a perception among many mathematicians that current AI tools are far from proving their theorems, or that the theorems being proven by AI are not of particular interest.

It took time to get some mathematicians interested in formalization.

Evangelism for (Auto)formalization

Let's assume that mathematics (auto)formalization is part of the future.

Some mathematicians feel that interactive theorem provers (ITPs) are not worth the effort.

There is a perception among many mathematicians that current AI tools are far from proving their theorems, or that the theorems being proven by AI are not of particular interest.

It took time to get some mathematicians interested in formalization.

It will likely take longer to generate interest in autoformalization.

Evangelism for (Auto)formalization

Let's assume that mathematics (auto)formalization is part of the future.

Some mathematicians feel that interactive theorem provers (ITPs) are not worth the effort.

There is a perception among many mathematicians that current AI tools are far from proving their theorems, or that the theorems being proven by AI are not of particular interest.

It took time to get some mathematicians interested in formalization.

It will likely take longer to generate interest in autoformalization.

It may also take time to get AI researchers interested in (auto)formalization, although hopefully less time.

Evangelism for Large-Scale (Auto)formalization

There are several large-scale formalization projects that involve teams of people with diverse expertise (e.g., LTE, FLT).

Evangelism for Large-Scale (Auto)formalization

There are several large-scale formalization projects that involve teams of people with diverse expertise (e.g., LTE, FLT).

Could a similar approach be beneficial for autoformalization?

Evangelism for Large-Scale (Auto)formalization

There are several large-scale formalization projects that involve teams of people with diverse expertise (e.g., LTE, FLT).

Could a similar approach be beneficial for autoformalization?

AITP could also stand for "Autoformalization in Theorem Proving."

References I

[1] L-K Hua. *Introduction to number theory*. Springer Science & Business Media, 2012.

The End