証明ステップの逆形式化とステップ間の構造解析を通じた形式証明の自然言語翻訳

服部 清志 [1] 松崎 拓也 [1] 藤原 誠 [1]
[1] 東京理科大学大学院 理学研究科 応用数学専攻
1424521@ed.tus.ac.jp matuzaki@rs.tus.ac.jp makotofujiwara@rs.tus.ac.jp

概要

自動形式化とは、自然言語で書かれた数学証明を機械検証可能な形式証明に自動翻訳する技術のことである。この技術は近年の大規模言語モデルの発展に伴って盛んに研究されているが、学習に用いる自然言語証明-形式証明ペアデータの不足が課題となっている。本研究では LLM の逆形式化能力を活用した形式証明の自然言語翻訳手法を提案する。また、自然言語証明の表現に沿って作成した形式証明データに対して本手法を適用し、生成された自然言語証明の品質を分析及び評価する。

1 はじめに

形式証明とは、コンピューター上で検証可能な形式言語によって表された数学証明のことを指す。形式証明は主に Isabelle [1] や Coq [2]、Lean [3] に代表される定理証明支援系の言語で記述される。

自動形式化とは、人間が書く証明 (自然言語証明) を形式証明の形に自動変換するタスクである。自動形式化技術は高コストな人手による形式化作業の負荷を軽減するだけでなく、正確な動作が要求されるシステムの自動検証 ([4] など) や自動定理証明の実現にも応用できる。近年の大規模言語モデル (Large Language Model; LLM) の発展により、機械翻訳タスクとしての自動形式化技術の研究は活発に進められている。しかし、モデルの学習データに用いる自然言語証明と形式数学のペアデータが不足していることや、自動形式化の結果を評価するための有効な指標がないことが課題になり十分な成果を上げることが出来ていないという実情がある。

本研究では、形式証明のステップベースの逆形式化、即ち形式証明から自然言語証明への方向の証明ステップごとの翻訳と、証明ステップ間の構造解析を踏まえた要約の 2 段階からなる、形式証明と自然言語証明のペアデータの作成手法を提案する。また、大学学部レベルの数学証明から証明構造に沿って形式化し作成された形式証明データに対して提案手法を適用し、生成結果の分析及び評価を行う。

2 先行研究

自動形式化及び逆形式化に関する近年の研究のうち、本研究と特に関連する研究を幾つか紹介する。Jiang ら [5] は、Isabelle および Lean で書かれた形式証明を GPT-4 を用いて逆形式化することで大規模な形式命題-自然言語命題ペアデータセットを構築する手法を提案した。我々の提案手法もプロンプトベースの逆形式化という点では一致しているが、Jiang らの手法は形式命題のみを対象とし、本研究は形式証明を対象としている点で異なる。

Gao ら [6] は Lean の形式証明ライブラリ中の形式命題及び形式証明から静的解析ツールを用いて抽出した情報を逆形式化プロンプトに活用することで高品質なペアデータセットを構築する手法を提案した。我々の研究もこの流れを踏襲するが、本研究は形式証明から得られる抽象構文木を用いて証明ステップ間の依存関係を解析し、それを逆形式化結果の要約に反映する。形式証明から得られる証明ステップ間の依存関係構造を適切に反映させることで、LLM に直接要約を生成させるよりも証明の論理構造を反映した高品質な自然言語証明を生成することができると考えられる。

3 定理証明支援系について

定理証明支援系とは、数学証明を記述するための形式言語と形式言語を用いて書かれた証明に対する自動検証システムを提供するソフトウェアを指す。

Lean [3] は依存型付きラムダ計算に基づく関数型プログラミング言語を提供する証明支援系の 1 つであり、「タクティク」と呼ばれるコマンドを用い

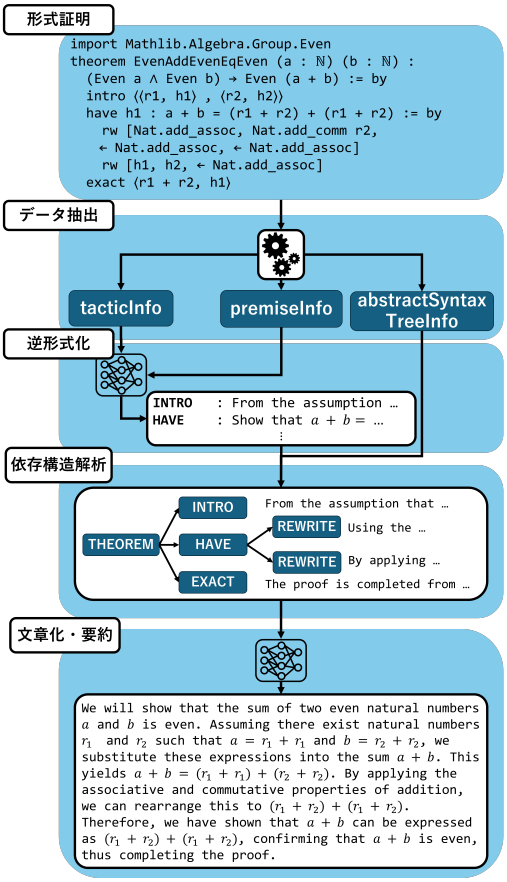

図 1　形式証明の自然言語翻訳手法

た対話型の証明支援機能を持つ。また、ユーザーコミュニティによって Lean 上で構築されている形式証明ライブラリ Mathlib を活用することで、形式証明を効率よく作成できる。本研究では 2021 年にリリースされた Lean4 及び Mathlib4 を提案手法の実装および few-shot 例の作成に使用している。

4　提案手法：形式証明の逆形式化

この節では、本論文で提案する形式証明翻訳手法について説明する。本手法は Lean4 で書かれた形式証明を、情報抽出 (4.1)、逆形式化 (4.2)、依存構造解析 (4.3)、要約 (4.4) の 4 つの工程で自然言語証明に翻訳する。図 1 に手法の概略を示す。

4.1　形式証明からの情報抽出

まず、逆形式化タスクで LLM に提供するために用いる形式証明の様々な情報を Lean のカーネルから抽出する。情報の抽出には LeanDojo [7] のデータ抽出ツールをベースとしたものを使用する。ここでは、以下の 3 つのデータを抽出する。

- **タクティク情報 (tacticInfo)**
  形式証明内で適用したタクティク、タクティクの適用前後の形式証明の状態等の情報を持つ。
- **前提情報 (premiseInfo)**
  形式証明中で使用されている定義・定理の名前やそれらがコーディングされているソースファイルへのパスなどの情報を持つ。
- **抽象構文木情報 (abstractSyntaxTreeInfo)**
  Lean のカーネル上で持っている形式証明の抽象構文木を JSON の配列形式で持つ。

4.2　形式証明の各証明ステップの逆形式化

抽出したタクティク情報と前提情報を用いて、形式証明の各証明ステップに対応する操作を説明する文を LLM を用いて生成する。

生成手順としては、まずタクティク情報から証明ステップに適用したタクティクと適用前後の証明状態 (仮定 ⊢ ゴールの形で表される) を取得する。

次に、適用したタクティクで用いた定理や定義を前提情報の中から探索し、ソースファイルから情報を抽出する。図 1 最上段の 3 つ目のタクティクである `rw[Nat.add_assoc, ... ←, Nat.add_assoc]` は、`Nat.add_assoc` と `Nat.add_comm` という 2 つの定理を使用している。そのため、これらの定理がコーディングされているソースファイルにアクセスし、それぞれの定義 (`add_assoc : ∀(n m k : Nat), (n + m) + k = n + (m + k)` および `add_comm : ∀ (n m : Nat), n + m = m + n`) を抽出する。

次に、生成時に LLM に与える few-shot 例を取得する。few-shot 例は Lean および Mathlib で定義されているタクティクの操作の種類ごとに作成されており、それぞれは適用したタクティクと適用前後の証明状態、そして出力となる操作説明文から成る。適用したタクティクと証明状態のデータは 4.1 で述べたツールを用いて、few-shot 例を作成するために用意した自作の形式証明及び Mathlib の形式証明から抽出したものを使用する。また、対応する出力例は、証明中に含まれる文として自然であり、与えられたタクティクがどのような操作に対応し、証明状態中のどの部分に注目して出力するかを LLM に対し明示することを意図してすべて手動で作成した。

図 2 は、背理法を用いることを宣言する `by_contra` タクティクに対する few-shot 例を示している。ここでは、目標である $b = 0$ という命題を否定した命題を仮定に加え、目標を矛盾を導くことに切り替えている。そのため、出力例は「b が 0 でないこと」を仮

```
goalsBefore
a b : ℤ
h : a + b * √2 = 0
⊢ b = 0
goalsAfter
a b : ℤ
h : a + b * √2 = 0
bh : ¬b = 0
⊢ False
appliedTactic
by_contra bh
output
Assume that b is nonzero
and show a contradiction.
```

図 2 few-shot 例

定して矛盾を導くことを明示する文章にしている。

最後に、ここまでの手順で取得した情報をプロンプトとともに LLM に与え、各証明ステップに対応する自然言語で書かれた説明文を生成させる。

ここまでの手順はタクティクを使用した全ての証明ステップに対して実行される。

4.3 証明ステップの依存構造解析

タクティク間の依存関係を解析し、前工程で作成した各証明ステップに対応する操作説明文を証明ステップ間の依存関係を表す木構造のノードに配置する。この木構造は、証明される命題を根とし、各証明ステップの説明文を子に持つ。また、証明ステップ内で中間ゴールとなる命題を証明する場合は、中間ゴールの宣言を行う証明ステップの説明文の子にその命題の証明ステップの説明文が来るようになっている。証明ステップ間の依存関係は形式証明から抽出された抽象構文木を解析することによって得られるタクティク間の親子関係から導かれる。

例えば、図 1 の例 (図中下から 2 つ目の囲み) では、仮定の導入 (intro)、中間ゴールの宣言 (have)、式変形による二つの等価性の証明 (rw)、命題の証明 (exact) の 5 つの操作が行われており、木構造は命題を根とし、命題の証明部分が部分木である高さ 2 の木構造に変換されている。

4.4 逆形式化データの要約

前工程で構築した木構造を活用し、全証明ステップの操作説明文を 1 つの証明になるよう要約する。

まず、木構造の根である形式命題を LLM を用いて逆形式化する。なお、ここでの逆形式化は 4.2 節で用いたものとは別のプロンプトを使用し、手動で作成した 3 つの few-shot 例と命題内で使用されている定義を情報として与えて生成している。

次に、形式証明の構造に沿った自然言語証明を生成するため、木構造に含まれる各部分木に対して部分証明の生成を行う。部分証明とは、部分木の根となるタクティクで宣言された中間ゴールの自然言語証明であり、その内容は根以外の部分木の節に対応する操作説明文の情報から作成される。また、部分証明の生成は再帰的に行われる。すなわち、部分木の中にさらに部分木が含まれていた場合は、含まれている方の部分証明をまず生成し、その生成結果を要約することで元の部分木の部分証明を生成する。

最後に、形式命題を逆形式化して得られた命題の証明文として、根の子要素すべての部分証明を用いて自然言語命題の証明文を生成する。

5 評価実験

5.1 実験設定

逆形式化及び要約には OpenAI の GPT-4o-mini (gpt-4o-mini-2024-07-18) を使用した。逆形式化タスクでは正確かつ few-shot の例文フォーマットに則った出力を得るため、生成時の temperature を 0.4 に設定した。また、要約タスクでは自然な証明文を生成させるため、生成時の temperature を 1.0 に設定した。

提案手法の評価には、宮島静雄著「微分積分学 I」[8] の 1.1 及び 1.2 の証明問題 17 個を証明の構造に沿って手動で形式化したもの [9] を使用した。

5.2 証明ステップごとの逆形式化の評価

逆形式化の出力に対する評価は、5.1 で述べた評価データに加え、形式証明中で使用する補題 21 個を加えた計 38 個の形式証明に対して提案手法を適用して作成した証明ステップの逆形式化結果 1265 個を対象に行った。

出力は誤った情報を含まないか、証明ステップの内容を十分に説明できているか、証明ステップの内容を説明する上で冗長な情報が含まれていないか、証明ステップの内容に関係のない情報が含まれていないか、形式言語の文法で書かれた式が含まれていないか、の 5 つの点で評価した。

評価結果を表 1 に示す。結果を見ると、まず、出力の約 7 割が逆形式化の出力として必要十分であることがわかる。これは、形式証明の証明ステップの逆形式化という新しいタスクに対しても、プロンプトと few-shot 例を工夫することで汎用 LLM により

表 1 逆形式化の評価結果。評価項目すべてを満たしたものを「正常」な出力として評価している。また、出力に問題があった場合は、該当する点すべてで個数を計上している。

	正常	誤情報	情報不足	不必要な言及	無関係な内容	翻訳漏れ
該当数	881	119	48	195	9	36
総数に対する割合 (%)	69.6	9.4	3.8	15.4	0.7	2.8

十分に対応できることを示している。

一方で、15%程度の出力に何らかの余分な情報が含まれていることがわかる。例えば、論理積 $A \wedge B$ を証明する状態から、A と B それぞれを示す状態に移行させる `apply And.intro` というタクティクに対して、“... This tactic effectively split the original goal into two distinct parts for clearer analysis and proof.” という、証明の流れというよりは Lean のタクティクの動作についての説明文を含んだ出力が得られた。この問題は、与える few-shot 例を改善し量を増やす、あるいはプロンプトにタクティクに対応する出力文のテンプレートを渡し、テンプレート中の穴あき部分のみを LLM に推論させて埋めるタスクに置き換える等の工夫によって改善されると考える。

また、出力の約 10%が証明ステップの内容を正しく反映できていなかった。例えば、ある条件 A に対し A か $\neg A$ かで場合分けを行う証明では、「A の場合について示す」という前者の場合分けの開始を表す文が出力されるべきところに「$\neg A$ の場合について示す」という後者の場合分けの開始を表す文が出力されることが多くあった。これは LLM が証明ステップの形式に適応できていないこと、コマンドの操作内容を出力に反映できていないことが原因として考えられる。この問題を解決するには、形式言語で訓練された LLM を用いる、プロンプトに与える情報量を増やすなどの工夫が必要であると考える。

5.3 要約に対する評価

要約に対する評価は、5.1 で述べた 17 個のペアデータを用いて行う。まず、17 個の自然言語証明に対して平均 6.4 個の評価項目を設定した。各評価項目は、証明中で示す命題や使用した定理、新しく導入する変数など、証明を構成する上で言及すべき要点を手動で設定した。次に、各自然言語証明に対応する形式証明に提案手法を用いて要約を生成し、生成結果が各評価項目について正しく言及できているかどうかを「正しく要点を押さえられている (○)」、「要点を部分的に押さえられている (△)」、「要点を押さえられていない (×)」の 3 段階で評価した。そして、生成結果のスコアを以下の式で計算した。

$$s = \frac{(\text{○の個数}) + 0.5 \times (\text{△の個数})}{\text{評価項目数}}$$

結果として、平均スコアは 0.7442、最大値は 1、最低値は 0.45、中央値は 0.75 であった。各証明に対する評価結果やスコアは付録 A に掲載する。

分析を行った結果として、まず、証明構造に沿った要約を行うことで、証明の流れが概ね反映された要約を生成できることが分かった。一方で、証明ごとの情報の粒度に大きなばらつきがあることが分かった。例えば「収束数列は有界である」という命題を用いて数列の絶対値が定数で上から抑えられる、という部分に対応する生成結果が “By the property of convergent sequences,...” という曖昧なものであった一方で、「正の実数集合 P の下限は 0 である」という命題の証明では、“P is non-empty by noting that $1 \in P$ since $1 > 0$.” というように、形式証明では必要なものの人間の書く証明においては自明なものとして省略される内容が反映されることがあった。

また、生成結果の品質に幅があることが分かった。同じ形式証明に対する生成でも、各要点を正しく押さえられている生成結果がある一方で、自然言語証明にも形式証明にも表れていない変数を勝手に導入している生成結果や計算の途中式が要約の段階で壊れてしまった生成結果があった。

これらの問題を解決するには、「情報の粒度」を制御する手法の開発や、より正確性が担保される要約手法を使用するなどの工夫が必要であると考える。

6 おわりに

本研究では、自動形式化技術の実現において課題となっている自然言語証明-形式証明ペアデータの構築を実現することを目的として、形式証明を証明ステップごとに逆形式化し文書要約することで自然言語証明を生成する手法を提案した。また自然言語証明の構造に沿って作成された形式証明に対して提案手法を適用し分析を行い、手法が有用なことを示し、また幾つかの課題点を示した。今後はより高品質な逆形式化生成手法の構築、証明文生成に適した要約手法の研究に取り組んでいきたい。

参考文献

[1] Lawrence C Paulson. Isabelle: The next 700 theorem provers. In **Logic and computer science**, Vol. 31, pp. 361–386. Citeseer, 1990.

[2] The Coq. The Coq Proof Assistant : Reference Manual : Version 7.2. Technical Report RT-0255, INRIA, February 2002.

[3] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, **Automated Deduction - CADE-25**, pp. 378–388, Cham, 2015. Springer International Publishing.

[4] Xavier Leroy. Formal verification of a realistic compiler. **Commun. ACM**, Vol. 52, No. 7, p. 107–115, July 2009.

[5] Albert Q. Jiang, Wenda Li, and Mateja Jamnik. Multi-language diversity benefits autoformalization. In **The Thirty-eighth Annual Conference on Neural Information Processing Systems**, 2024.

[6] Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated lean 4 dataset. **arXiv preprint arXiv:2410.10878**, 2024.

[7] Kaiyu Yang, Aidan M Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In **Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track**, 2023.

[8] 宮島静雄. 微分積分学 I - 1 変数の微分積分 -. 共立出版, 2010.

[9] 服部清志. 松崎拓也. 藤原誠. Autoformalization に向けた自然言語証明構造の形式化. 言語処理学会全国大会論文集, 2024.

A　要約に対する評価結果詳細

今回の実験対象である証明 17 個の評価結果を表 2 にまとめた。定理 1.1 から定理 1.6 が 1.1 節 (連続性の公理)、定理 1.7 から定理 1.9 が 1.2 節 (数列の収束の $\varepsilon - n_0$ 式定義) の命題である。1.1 節の問題に対する生成結果の平均スコアは 0.71、中央値は 0.71 であった。1.2 節の問題に対する生成結果の平均スコアは 0.78、中央値は 0.78 であった。この結果からみると、問題の種類にかかわらず、出力の品質はおおよそ同じであることがわかる。

	○の数	△の数	×の数	スコア
定理 1.1	2	5	1	0.56
定理 1.2	6	1	2	0.72
補題 1.3 (1)	5	1	0	0.92
補題 1.3 (2)	2	3	0	0.70
補題 1.3 (3)	2	0	1	0.67
定理 1.4	6	1	1	0.81
系 1.5	1	0	1	0.50
定理 1.6	5	1	1	0.79
定理 1.7	4	1	1	0.75
定理 1.8 (1) (a)	3	0	0	1.00
定理 1.8 (1) (b)	6	0	0	1.00
定理 1.8 (1) (c)	3	2	0	0.80
定理 1.8 (1) (d)	3	3	2	0.56
定理 1.8 (1) (e)	4	1	5	0.45
定理 1.8 (2) (a)	8	1	0	0.94
定理 1.8 (2) (b)	7	0	2	0.78
定理 1.9	3	1	1	0.70

表 2　要約結果に対する評価。Ground-truth である自然言語証明から証明内で言及すべき要点を複数個列挙し、要約結果が各要点を満たしているかを 3 段階評価している。スコアは (○の数 + 0.5(△の数))/要点数合計 で計算し、小数点第 3 位を四捨五入している。

B　生成結果及び評価項目の具体例

以下の例は、宮島静雄著「微分積分学 I」[8] の定理 1.8 (2)(a)「$\{x_n\}_n$, $\{y_n\}_n$ が収束し, すべての n で $x_n \leq y_n$ ならば, $\lim_{n\to\infty} x_n \leq \lim_{n\to\infty} y_n$ が成り立つ.」という命題に対する元の証明文、その証明の構造に沿って作成した Lean の形式証明に提案手法を適用して逆形式化したもの (生成結果)、生成結果に対する評価項目及び実際の評価である。

B.1　元の証明文

$a := \lim_{n\to\infty} x_n, b := \lim_{n\to\infty} y_n$ とする. このとき任意の $\varepsilon > 0$ に対して $n_1, n_2 \in N$ があって, $n \geq n_1 \Rightarrow |x_n - a| < \frac{\varepsilon}{2}, n \geq n_2 \Rightarrow |y_n - b| < \frac{\varepsilon}{2}$ となる. n_1, n_2 の大きいほうを n_0 とすれば, $n \geq n_0$ のとき

$$a - \frac{\varepsilon}{2} < x_n \leq y_n < b + \frac{\varepsilon}{2}$$

なので $a < b + \varepsilon$ となる. $\varepsilon > 0$ は任意だったので, これから $a \leq b$ が分かる (補題 1.3 参照).

なお、証明文中に登場する「補題 1.3 参照」は、同書籍中の補題 1.3(2)「$a, b \in R$ が任意の $\varepsilon > 0$ に対して $a < b + \varepsilon$ をみたしていれば $a \leq b$.」という命題を指す。

B.2　生成結果

We will show that if the sequences x and y converge to limits a and b, respectively, and if $x_n \leq y_n$ for all natural numbers n, then $a \leq b$. Let a and b denote the limits of the sequences x_n and y_n respectively. For any positive ϵ, we select natural numbers n_1 and n_2 such that for all $n \geq n_1$, $|x_n - a| < \frac{\epsilon}{2}$ and for all $n \geq n_2$, $|y_n - b| < \frac{\epsilon}{2}$. Let $n_0 = \max(n_1, n_2)$; then for all $n \geq n_0$, we have $a - \frac{\epsilon}{2} < x_n \leq y_n < b + \frac{\epsilon}{2}$. This leads to the inequalities $a - \frac{\epsilon}{2} < b + \frac{\epsilon}{2}$. Rearranging this expression gives $a < b + \epsilon$. Since this must hold for any positive ϵ, we conclude by taking the limit as ϵ approaches 0 that $a \leq b$. Thus, we have established that the limit of the sequence x is less than or equal to the limit of the sequence y, completing our proof.

B.3　評価項目及び評価結果

- x_n の極限を a、y_n の極限を b とすること ⇒ ○
- ε を任意にとること ⇒ ○
- $n_1 \in N, n \geq n_1 \Rightarrow |x_n - a| < \frac{\varepsilon}{2}$ の明示 ⇒ ○
- $n_2 \in N, n \geq n_2 \Rightarrow |y_n - b| < \frac{\varepsilon}{2}$ の明示 ⇒ ○
- n_1, n_2 の大きいほうを n_0 と定義すること ⇒ ○
- $n \geq n_0, a - \frac{\varepsilon}{2} < x_n \leq y_n < b + \frac{\varepsilon}{2}$ の明示 ⇒ ○
- $a < b + \varepsilon$ が導かれることの明示 ⇒ ○
- 補題 1.3 を用いることの明示 ⇒ △
- $a \leq b$ となることの明示 ⇒ ○