# An in-depth human study of the mathematical reasoning abilities in Large Language Models

Carolina Dias-Alexiou, Edison Marrese-Taylor, Yutaka Matsuo
Graduate School of Engineering, The University of Tokyo
{carolina.dias,emarrese,matsuo}@weblab.t.u-tokyo.ac.jp

## Abstract

We study the generalization capabilities of Large Language Models through the lens of mathematical reasoning, asking if these models can recognize that two structures are the same even when they do not share the same nomenclature. We propose a human study to evaluate if LLMs reproduce proofs that they have most likely seen during training, but when the symbols do not match the ones seen. To test this in a controlled scenario, we look at proofs in *propositional calculus*, foundational for other logic systems, semantically complete and widely discussed online. We replace the implication operator ($\rightarrow$) with an unrelated, arbitrary symbol ($\spadesuit$) and ask experts to evaluate how the output of a selection of LLMs changes in terms of compliance, correctness, extensiveness and coherence. Our results show that nearly all our tested models produce lower quality proofs in this test, in particular open-weights models, suggesting the abilities of these LLMs to reason in this context have important limitations.

## 1 Introduction

Mathematical reasoning is a key aspect of human intelligence that encompasses pattern recognition and logical entailment. The development of artificial intelligence systems capable of tasks such as solving applied and theoretical mathematical problems has been a long-standing focus of research in the fields of machine learning and natural language processing, dating back to the 1960s [1, 2].

Our interest in this topic rises from recent attempts to use language models for theorem proving by means of ITPs' programming languages and databases of theorems with their proofs. Deep learning models can be trained in one of these many programming languages, and then used to generate mathematical proofs. Data sources for neural theorem proving in ITPs include interactive learning environments that interface with ITPs, and datasets derived from proofs in ITP libraries.

When it comes to mathematical reasoning, and in particular to the ability of models to understand logical statements, we note that despite the abundance of studies, previous works generally assume that only "variables" are the ones not constant throughout problems. However, we see that in mathematics different nomenclatures are used in different areas, as well as in different time periods, to express the same ideas. Examples of this fact include the use of $\supset$ instead of $\rightarrow$ in logic, particularly in texts written before computers; Leibniz's $\frac{\mathrm{d}f}{\mathrm{d}x}$, Lagrange's $f'$, and Newton's $\dot{f}$ notation in differential calculus; the different notation for the inner product for physicists $\langle\cdot|\cdot\rangle$ and for mathematicians $\langle\cdot,\cdot\rangle$; prefix $f(x)$ and postfix $(x)f$ notations for functions; and different notations for the von Neumann generated algebra, such as: $W^*(\cdot,\cdot)$ and $\cdot \vee \cdot$, to name a few.

As we attempt to use LLMs to tackle proof generation tasks, for example using the "informal theorem proving" approach (please see §2 for more details on this), we think researchers and practitioners need to take this fact into consideration. Furthermore, and in contrast to all these models, current state-of-the-art models in NLP are trained on large datasets of text extracted from the web. In this case, we have limited or no control on the kinds of expressions and/or operators that the models are exposed to during training. We can, however, still assume the models have been exposed mostly to the standard nomenclature.

Given this scenario, we ask: *can these models recognize that two structures are the same even if they do not share the same nomenclature?* For example, can models reproduce proofs that they have most likely seen during training, but when the symbols do not match the ones seen? If so, to what extent are these proofs plausible and correct? In order

to answer these questions, we perform an in-depth human evaluation to assess the quality of the output generated by LLMs when prompted to generate proofs similar to the ones seen during training but with expressions that use a different notation.

## 2 Related Work

**Automated Theorem Proving** The first proofs using computers started appearing as early as the 1950s, soon after electronic computers became available. This played a big role in the development of the field of automated reasoning, which itself led to the development of AI. Most of the early work on computer-assisted proof was devoted to *automated theorem proving* (ATP) [3], in which machines were expected to prove assertions fully automatically. The increased availability of interactive time-sharing computer operating systems in the 1960s allowed the development of *interactive theorem provers* (ITPs) in which the machine and the user work together to produce a formal proof. While ATPs include proof-search algorithms to generate whole proofs, ITPs usually check the validity of human input statements, although they may also include reduced automated tools.

More recently, work on "informal" theorem proving has presented an alternative medium for theorem proving, in which statements and proofs are written in a mixture of natural language and symbols used in "standard" mathematics (e.g., in LaTeX), and are checked for correctness by humans. Here we find the work of [4] who developed NaturalProofs, a large-scale dataset of 32k informal mathematical theorems, definitions, and proofs, and provided a benchmark for premise selection via retrieval and generation tasks. Most of the data is taken from websites like `proofwiki.com`, and though this enables more flexibility when proving, the task is approached in a way similar to ITPs.

**Mathematical Reasoning in LLMs** Early works attempting to study the ability of models to recognize patterns in mathematical expressions focused on the manipulation of simple expressions using standard notation. For example, [5] trained models on datasets in which pairs of examples contain Boolean logic and arithmetic expressions which are known to be equivalent. For example, expressions like $c^2$ and $(c \cdot c) + (b - b)$ are equivalent. However, expressions with the same structure, but different variables, such as $c \cdot (a \cdot a + b)$ and $f \cdot (d \cdot d + e)$, are not. They showed that such models were capable of relating non-paired expressions, like $a - (b - c)$ and $b - (a + c)$, as negations of each other.

[6] studied the ability of neural networks to understand logical entailment via training models on synthetic datasets of logical statements and their evaluations (True/False). Concretely, they generated datasets of triples of the form $(A, B, A \vDash B)$, where $A$ and $B$ are formulas of propositional logic, and $A \vDash B$ is 1 if $A$ entails $B$, and 0 otherwise. They concluded that, from the models available at the time, those with a tree structure seemed to be better for domains with unambiguous syntax.

In these works, expressions are generated automatically starting from a set of simple rules plus a set of arbitrary combinations. This allows to scale and control the types of expressions that are shown to the model during training/inference. Later works, like [7] and the more recent [8] shift to a question-answer format using natural language, with the release of the GSM8K and GPQA datasets, respectively, while also extending the class of questions to other areas of mathematics, like calculus and probability, where the ability to control for the type of expressions is reduced.

## 3 Proposed Approach

To study the posed research questions under a controlled scenario, we look at proofs in *propositional calculus*, a branch of formal logic that deals with propositions, which can be true or false, and relations between propositions, including the construction of arguments based on them [9]. Propositional calculus, also known as zeroth-order logic, does not deal with quantifiers over non-logical objects (unlike first-order or higher-order logic). There are several reasons that we think make this the ideal scenario for our study: (1) All the machinery of propositional logic is included in first-order logic, higher-order logic, and all mathematics. In this sense, propositional logic is the foundation of other logic systems; (2) Propositional calculus is semantically complete, i.e. any tautology (true formulas) can be proved with the formal axioms and the rules of inference of the system, and (3) Being the subject of common undergraduate courses, demonstrations in this context have been widely discussed online so we can reasonably assume that LLMs have been exposed to these type of proofs.

Propositional calculus is typically studied with a formal system, which contains a formal language and a deductive system. The language is composed of a set of well-formed formulas, which are strings of symbols from an alphabet (composed of propositional variables and propositional connectives) formed by a formal grammar (formation rules). The deductive system, in turn, contains the rules of inference, a function which takes premises and returns conclusions. To assess how models generalize in this scenario, we compared proofs generated by these models using 'usual' and 'unusual' symbols for connectives.

We use a standard proof system usually referred to as a Hilbert system. It is a deductive system that generates theorems from axioms (a tautology taken as starting point for further reasoning) and *modus ponens*. *Modus ponens* can be summarized as: If P implies Q and P is known to be true, one can conclude that Q must also be true. It is generally expressed as $\{P \rightarrow Q,\ P\} \vdash Q$, where the turnstile symbol ($\vdash$) denotes derivability, i.e. there is a formal derivation of a theorem from the axioms. As for connectives, we limit it to the logical and ($\wedge$), logical or ($\vee$), the negation operator ($\neg$), and the implication operator ($\rightarrow$). For the axioms, we use a common set of 14 axioms used in undergraduate courses, shown in Figure 1 in our supplementary material (§A).

For our study we propose to replace the implication operator ($\rightarrow$) with an unrelated, arbitrary symbol (♠). In order to produce a significant perturbation in the input token distribution, we specifically select the unicode representation of the symbol (U+2660) for the replacement. Alternative replacements are left for future work. We select two common theorems from propositional calculus extracted from [10], shown in Figure 2 and Figure 3, in §A, and test models in two different scenarios, as follows.

**Full Context (FC)** Our first evaluation scheme is intended to simulate a noisy retrieval step, prior to the proof generation. Concretely, we offer the model the complete set of axioms together with the selected rule of inference, *modus ponens*.

**Selected Context (SC)** We assume that the relevant axioms for the requested proof have already been selected by an oracle, and we offer only these axioms and rule of inference to the model input. For each question, we manually select the axioms required (Axioms 6, 7, 10 for Question 1; Axioms 7, 8 for Question 2).

A key point of our study is to ensure that the generated proofs are checked by mathematicians. Previous work has stressed the need to rely on experts for evaluation of theorem proving systems, including [11], who carry on an in-depth annotation where an expert annotator is presented with the theorem, proof-so-far, and a generated next-step. [12] also highlight that human evaluation of advanced mathematics that approaches research level is expensive and requires experts. The evaluation of the output of the language model for their work was performed by the authors, who are all mathematicians.

To perform the evaluation, we concretely rely on one volunteer (one of the authors of this paper) who has a degree in mathematics. We design an annotation interface where for each case, we show the annotators the exact input fed to the model, as well as the output generated. Below, we list the tasks that we require our annotators to perform.

- First, we ask the annotators if the output of the model contains a proof or not.
- We present the steps of the correct proof and ask the annotators to judge if each step appears in the output of the model. Additionally, if the step invokes an axiom or rule of inference, we ask them to do the following.
  - If the step invokes an axiom, to check if variables were substituted correctly.
  - If the step uses a rule of inference, we ask them to check if the rule is properly invoked.
  - To judge whether the step contributes to the proof in the sense that it stirs the overall flow of the proof in the right direction towards conclusion.
- With respect to the coherence of the overall text, we ask the annotators to rate the output in a scale of 0 to 4 via the following labels: "Very Incoherent, Incoherent, Neither Coherent nor Incoherent, Coherent, Very Coherent".

# 4 Results

For this study, we consider the following models: (1) API-based LLMs, including ChatGPT (*gpt-3.5-turbo-0125*) [13], Claude 3 Opus (*claude-3-opus-20240229*) [14], (2) Open-weights models, including Llama 3 (*Meta-Llama-3-8B-Instruct*) [15, 16], Llama 3.1 (*Llama-3.1-8B-Instruct*) [17] and Gemma 2 (*gemma-2-9b-it*) [18]. The

**Table 1** Summary of the results of our human evaluation study, where Ctx. is short for context. Bold numbers indicate the best score for each pair of (→, ♠) prompts for a given model, and we highlight the best score of all across the FC and SC scenarios for each model.

| Model | Ctx. | Compliance | | Extensiveness | | Correctness | | Flow | | Coherence | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | → | ♠ | → | ♠ | → | ♠ | → | ♠ | → | ♠ |
| ChatGPT | SC | 100% | 100% | **62.38%** | 49.52% | **24.29%** | 5.71% | **24.76%** | 17.14% | **2.67** | 2.33 |
| | FC | 100% | 100% | **36.19%** | 30.48% | **7.14%** | 4.76% | **20.95%** | 12.86% | **3.00** | 2.60 |
| Claude 3 Opus | SC | 100% | 100% | **95.24%** | 93.14% | **76.19%** | 60.57% | **80.95%** | 70.29% | **3.83** | 3.40 |
| | FC | 100% | 100% | **85.71%** | 75.71% | **60.95%** | 27.62% | **71.43%** | 41.43% | **3.67** | 3.00 |
| Gemma 2 (9B) | SC | 33.33% | 0% | **11.90%** | - | 0% | - | **2.38%** | - | **4.00** | - |
| | FC | 50.00% | 0% | **16.67%** | - | 0% | - | **2.38%** | - | **4.00** | - |
| Llama 3 (8B) | SC | 100% | 83.33% | **49.52%** | 41.43% | **2.38%** | 2.38% | **11.90%** | 4.76% | **1.83** | 1.80 |
| | FC | 100% | 83.33% | **35.71%** | 21.90% | **9.05%** | 2.38% | **11.43%** | 0% | **2.17** | 1.60 |
| Llama 3.1 (8B) | SC | 100% | 100% | **42.38%** | 40.95% | 0% | **2.38%** | **4.76%** | 5.71% | **1.17** | **1.17** |
| | FC | 100% | 100% | **32.38%** | 24.29% | **3.33%** | 0% | **5.71%** | 4.76% | **2.00** | 1.50 |

latter models are obtained from HuggingFace, and quantized to 4-bits [19] to fit our GPU memory. For each input, we obtain 3 outputs from each model using a different random seed. We compute the following metrics to summarize model behavior.

- Percentage of times the model generated output that contains a proof, which we consider a measure of model compliance to our instruction (Compliance).
- Percentage of steps from the actual proof that appear in the generated proof, or to which extent it used the needed axioms and *modus ponens* (Extensiveness).
- Percentage of steps that appear in the generated proof and were correctly applied. In other words, the steps that are correct (Correctness).
- Percentage of steps that appear in the output and provide a expression which is a step towards finalizing the proof, even if it was not correctly deduced (Flow).
- Average coherence score reported for the overall text output from a given model (Coherence).

Table 1 summarizes the results of our evaluation efforts. We can clearly see that most models (with the exception of Gemma 2) show decreased performance across all metrics when prompted with (♠). Furthermore, most models (with the exception of Llama 3.1) showed decreased performance in the Full Context compared to the Selected Context. Most interestingly, the score for Flow was generally much higher than the Correctness, indicating that the model 'remembers' the right answer, even if it does not 'remember' how to deduce it.

The model with the best performance across all measures was Claude 3 Opus. Like for most models, its Extensiveness measure was subject to a decrease when one compares the easiest scenario (SC with →) to the hardest scenario (FC with ♠), but its lowest value (75.71%) was still higher than the highest Extensiveness measure of any other model. However, it did experience a sharp decrease in both Correctness (from 76.19% to 27.62%) and Flow (from 80.95% to 41.43%), the largest different performance of any of the measures. As table shows, there is also a big difference in performance between the API-based LLMs and the Open-weights models. The Gemma 2 model refused to provide a proof in most case, as it can be seeing by its Compliance measure, making it impossible to draw conclusions about its abilities. We think this suggests open-weights models are significantly behind APIs, which is well-aligned with performance measured in popular automatic benchmarks.

## 5 Conclusions

This paper studies the generalization capabilities of LLM through the lens of mathematical reasoning. We perform an in-depth human evaluation of the output of LLMs when they are prompted to produce basic proofs in propositional calculus, comparing their answers when we replace the implication operator (→) with an unrelated, arbitrary symbol (♠). Our results show that nearly all our tested models produce lower quality proofs in this test, in particular open-weights models, suggesting the abilities of these LLMs to reason in this context have important limitations. For future work we would like to extend this study to incorporate more proofs, models, and multiple annotators. We would also like to analyze how models react to other input perturbations, for example using other replacement symbols, and/or alternative representations for them.

# References


[1] Edward A Feigenbaum, et al. **Computers and thought**. McGraw-Hill, 1963.

[2] Daniel G Bobrow. Natural language input for a computer problem solving system. **AI Technical Reports**, 1964.

[3] John Harrison, Josef Urban, and Freek Wiedijk. History of Interactive Theorem Proving. In **Handbook of the History of Logic**, Vol. 9, pp. 135–214. Elsevier, 2014.

[4] Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. Naturalproofs: Mathematical theorem proving in natural language. In **Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track**, 2021.

[5] Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. Learning Continuous Semantic Representations of Symbolic Expressions. In **Proceedings of the 34th International Conference on Machine Learning**, pp. 80–88. PMLR, July 2017.

[6] Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. Can Neural Networks Understand Logical Entailment? In **International Conference on Learning Representations**, February 2018.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. **arXiv preprint arXiv:2110.14168**, 2021.

[8] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, November 2023.

[9] Propositional Logic | Internet Encyclopedia of Philosophy.

[10] Dino Rossegger. **Introduction to Mathematical Logic**. 2019.

[11] Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. In **Advances in Neural Information Processing Systems (NeurIPS)**, 2022.

[12] Simon Frieder, Martin Trimmel, Rashid Alawadhi, and Klaus Gy. LLM vs ITP. In **The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23**, October 2023.

[13] Greg Brockman, Atty Eleti, Elie Georges, Joanne Jang, Logan Kilpatrick, Rachel Lim, Luke Miller, and Michelle Pokrass. Introducing chatgpt and whisper apis. OpenAI Blog, 2023.

[14] Anthropic. Introducing claude. Anthropic, 2023.

[15] Aaron Grattafiori and the Llama 3 Team. The llama 3 herd of models, 2024.

[16] AI@Meta. Llama 3 model card. 2024.

[17] Introducing Llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/.

[18] Gemma Team. Gemma 2: Improving open language models at a practical size, 2024.

[19] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, **Advances in Neural Information Processing Systems**, Vol. 36, pp. 10088–10115. Curran Associates, Inc., 2023.

# A Appendix

**Prompt (portion)**

$$(\text{Ax 1})((\phi \wedge \psi) \to \phi)$$
$$(\text{Ax 2})((\phi \wedge \psi) \to \psi)$$
$$(\text{Ax 3})(\phi \to (\psi \to (\phi \wedge \psi)))$$
$$(\text{Ax 4})(\phi \to (\phi \vee \psi))$$
$$(\text{Ax 5})(\phi \to (\psi \vee \phi))$$
$$(\text{Ax 6})((\phi \to \chi) \to ((\psi \to \chi) \to ((\phi \vee \psi) \to \chi)))$$
$$(\text{Ax 7})(\phi \to (\psi \to \phi))$$
$$(\text{Ax 8})((\phi \to (\psi \to \chi)) \to ((\phi \to \psi) \to (\phi \to \chi)))$$
$$(\text{Ax 9})((\phi \to \psi) \to ((\phi \to \neg\psi) \to \neg\phi))$$
$$(\text{Ax 10})(\neg\phi \to (\phi \to \psi))$$
$$(\text{Ax 11})(\phi \vee \neg\phi)$$
$$(\text{Ax 12})((\phi \wedge \neg\phi) \to \psi)$$
$$(\text{Ax 13})((\phi \to (\psi \wedge \neg\psi)) \to \neg\phi)$$
$$(\text{Ax 14})(\neg\neg\phi \to \phi)$$
$$(\text{Modus Ponens})\{P \to Q, P\} \vdash Q$$

**Figure 1** Portion of the prompt provided to the LLMs showing the content of the full context provided, namely, the axioms and rules of inference we allow the models to use.

**Question**

**Question:** Prove that $\vdash ((\neg P \vee Q) \to (P \to Q))$.
**Answer:**

$$((\neg P \to (P \to Q)) \to ((Q \to (P \to Q)) \to ((\neg P \vee Q) \to (P \to Q)))) \quad (\text{Ax 6})$$
$$(\neg P \to (P \to Q)) \quad (\text{Ax 10})$$
$$((Q \to (P \to Q)) \to ((\neg P \vee Q) \to (P \to Q))) \quad 1, 2 \text{ MP}$$
$$(Q \to (P \to Q)) \quad (\text{Ax 7})$$
$$((\neg P \vee Q) \to (P \to Q)) \quad 3, 4 \text{ MP}$$

**Figure 2** Details of the first question (Question 1) utilized for our study, also showing the actual proof which we allow our annotators to see.

**Question**

**Question**: Prove that $\{(P \to Q), (Q \to R)\} \vdash (P \to R)$.
**Answer**:

$$(P \to Q) \quad (\text{hyp})$$
$$(Q \to R) \quad (\text{hyp})$$
$$((Q \to R) \to (P \to (Q \to R))) \quad (\text{Ax 7})$$
$$(P \to (Q \to R)) \quad 2, 3 \text{ MP}$$
$$((P \to (Q \to R)) \to ((P \to Q) \to (P \to R))) \quad (\text{Ax 8})$$
$$((P \to Q) \to (P \to R)) \quad 4, 5 \text{ MP}$$
$$(P \to R) \quad 1, 6 \text{ MP}$$

**Figure 3** Details of the second question (Question 2) utilized for our study, also showing the actual proof which we allow our annotators to see.