LEAN4PHYSICS: COMPREHENSIVE REASONING FRAMEWORK FOR COLLEGE-LEVEL PHYSICS IN LEAN4

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Lean4PHYS, a comprehensive reasoning framework for college-level physics problems in Lean4. Lean4PHYS includes *LeanPhysBench*, a college-level benchmark for Lean4 formal physics reasoning, which contains 200 hand-crafted and peer-reviewed statements formalized from university textbooks and physics competition problems. To establish a solid foundation for formal reasoning in physics, we also launch *PhysLib*, a community-driven repository that includes fundamental unit systems and theorems essential for formal physics reasoning. Based on the benchmark and Lean4 repository we composed in Lean4PHYS, we report baseline results using major expert Math Lean4 provers and state-of-the-art closed-source models, and provide an analysis of their performance. In the experiment, we identify that most expert provers do not outperform general models as they did in the math domain. This indicates a potential overfitting of the math domain rather than learning formal reasoning. We also conduct a comprehensive experiment showing that with *PhysLib* in the context, LLMs' performance on *LeanPhysBench* increases by **11.88%** on average, proving the effectiveness of our repository in assisting LLMs to solve the Lean4 physics problem. To the best of our knowledge, we are the first study to provide a physics benchmark in Lean4.

1 INTRODUCTION

Formal thinking capability has always been considered a cornerstone of human intelligence and a key objective of machine learning. With the emergence of Large Language Models (LLMs), many studies explore diverse ways to apply LLMs to perform various reasoning tasks. This includes general reasoning (Wang et al., 2024b; Suzgun et al., 2022; Talmor et al., 2018), math reasoning (Hendrycks et al., 2021; Cobbe et al., 2021; Guo et al., 2025), natural science reasoning (Saikh et al., 2022; Edwards et al., 2025), and many other domains. However, most works handle reasoning as a pure Natural Language (NL) task that relies on the answer checking to judge the correctness of reasoning, while being unable to verify the intermediate reasoning steps.

To make the reasoning process verifiable, researchers attempted to ground the reasoning procedure in formal logical systems, enabling the automatic verification of both the reasoning process and its results through Formal Languages (FLs). Through this idea, many FLs are developed, including Lean (De Moura et al., 2015; Moura & Ullrich, 2021), Coq (Coq, 1996), Isabelle (Paulson, 1994), and HOL (Harrison, 2009). Among them, Lean4 has received superior attention from both the academic and industry in recent days, making it one of the most well-studied formal languages in recent years. There are numerous benchmarks (Zheng et al., 2021; Gulati et al., 2024; Azerbayev et al., 2023), dataset (Dong & Ma, 2025; Wang et al., 2024a; Ying et al., 2024), and provers (Polu et al., 2022; Wang et al., 2024a; Xin et al., 2024; Wang et al., 2025c; Lin et al., 2025b; Dong & Ma, 2025; Xin et al., 2025; Ren et al., 2025) have been proposed.

However, current studies in formal reasoning primarily focus on the mathematical domain, leaving other domains that can also be formalized, such as physics, largely understudied. Moreover, most of the state-of-the-art expert provers claim their capability in formal reasoning by showing superior results in Lean4 math benchmarks. This raises concerns about whether such formal capability can

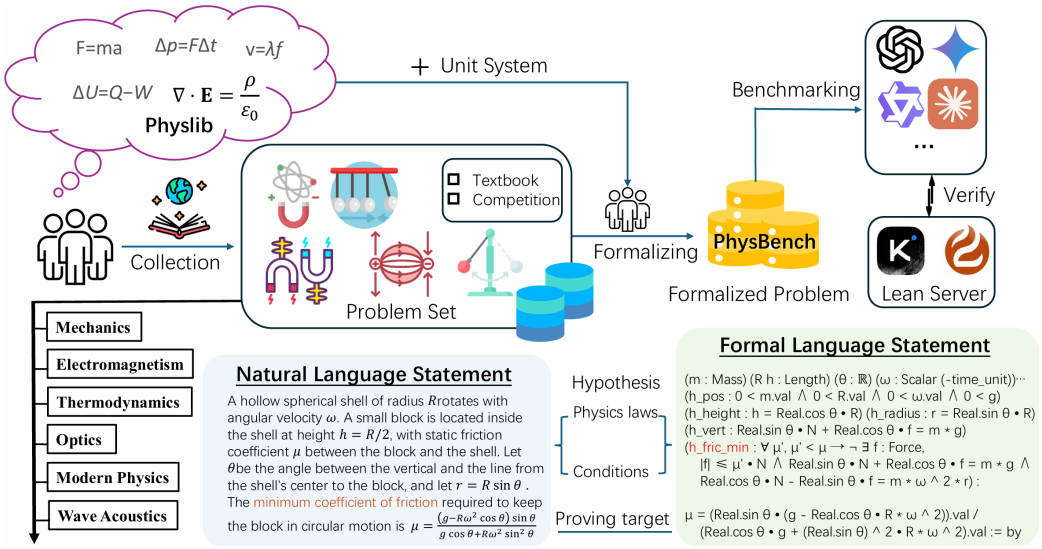

Figure 1: Overview of the Lean4PHYS framework: it consists of two components for supporting LLM formal physics reasoning. (1) *PhysLib*: an extensible repository providing a physics unit system and commonly used theorems. (2) *LeanPhysBench*: a benchmark of 200 hand-crafted theorems from high school competitions to elementary college level, designed to evaluate LLMs' Lean4 physics reasoning.

be transferred to other domains, like physics problems represented using similar Lean syntax, or current provers overfit to mathematical reasoning.

To answer the problems above, we propose Lean4PHYS, a comprehensive reasoning framework for college-level general-domain physics problems in Lean4. Lean4PHYS aims to provide a foundation for LLM-based formal physics reasoning in Lean4. The framework launches *PhysLib*, a foundation repository that supports formal physics reasoning. It includes a systematic treatment of physical units through a dedicated *UnitsSystem*, enabling dimensionally consistent symbolic computation, along with a growing collection of extendable formalized physics theorems. Based on the *PhysLib*, we develop *LeanPhysBench*, a college-level benchmark to evaluate LLMs' performance in Lean4 physics reasoning. The benchmark contains 200 manually-crafted Lean4 formal theorems whose difficulty level varies from standard university exercises to Olympiad-style competition problems. We construct the benchmark by systematically transforming natural-language problems into Lean4 theorems: extracting key conditions and relevant physical laws, defining explicit proving targets, restating the problems in logical form, and integrating them into verifiable Lean4 statements. To the best of our knowledge, *LeanPhysBench* is the first benchmark that evaluates LLMs' formal physics reasoning capability.

We summarize our contributions as follows:

1. We introduce *Lean4PHYS*, a comprehensive Lean4-based framework for formal physics reasoning. It provides *PhysLib*, a modular physics library supporting unit-aware calculations and extensible physical theorems.
2. Our framework innovates by bridging natural-language physics problems with formal Lean4 representations, enabling LLMs to learn domain-specific laws and reasoning patterns beyond standard math-oriented theorem provers.
3. We perform extensive experiments by applying leading models to *LeanPhysBench*. The experiments suggest that all current expert math provers and general models, regardless of their size, achieve suboptimal performance. Furthermore, we demonstrate that after integrating *PhysLib*, the models exhibit consistent performance enhancements. The results also indicate the potential for overfitting to the math dataset for the current expert Lean provers. When testing only on the hard dataset, we found that all models perform properly, indicating the current limitations for formal physics, especially statements involving complex mathematical operations such as integrals and derivatives.

Moreover, to the best of our knowledge, Lean4PHYS is the first work that tries to extend the LLM-based formal reasoning from math to a more general domain, which offers a new direction that attempts to formalize progressively more subjects.

2 THE LEAN4PHYS FRAMEWORK

In this section, we introduce the design and implementation of the Lean4PHYS framework in detail. The core idea of our framework is to provide a foundation and then evaluate the LLMs' formal physics reasoning capabilities. We firstly initiate *PhysLib*, a community collaborative foundation repository for Lean4 physics reasoning in Section 2.1. Subsequently, we present the details of how we construct *LeanPhysBench*, (to the best of our knowledge) the first benchmark for formal physics reasoning.

2.1 PHYSLIB

In this paper, we introduce *PhysLib*, a community-driven repository designed to support rigorous and machine-verifiable Lean4 formal physics reasoning. We build the library in a bottom-up manner for both the conceptual level of knowledge and the technical level of implementation. The current version of *PhysLib* contains the foundation of physics unit systems (Section 2.1.1), practical theorems to use in proving(Section 2.1.2), and a guideline for community development and extension (Section 2.1.3).

2.1.1 FOUNDATION OF PHYSICS: UNIT SYSTEM

Following the bottom-up design principle of *PhysLib*. The central challenge of formalizing physics from scratch is to identify the unchanged kernel that supports all the reasoning in the field. Unlike mathematics, where every basic building block is purely based on definition, physics, as well as other natural sciences, are primitively supported by empirical rules induced from experiments, which are naturally not as clearly defined as mathematics. For physics, we identify such a kernel as the unit system. Thus, we lay the foundation for physics reasoning by establishing the unit system.

In the implementation, we build the unit system by extending Mathlib Community (2019) with the International System of Units (SI) following the Tao (2025). The system contains seven basic units, including time, length, mass, electric current, thermodynamic temperature, amount of substance, and unit of luminous intensity Newell et al. (2019). Based on the unit system foundation, we introduce the first-order derivative over the most general and commonly used sections, namely, length, taking the derivative over time, introducing velocity.

2.1.2 TOPIC-BASED THEOREM DEVELOPMENT

Building upon the cross-topic kernel foundation system of units, we introduce topic-based theorem systems according to different needs in specific kinds of problems. Specifically, the current version of *PhysLib* splits problems into six major topics, namely: mechanics, waves & acoustics, thermodynamics, electromagnetism, optics, and modern physics. The topic split is inspired by Halliday et al. (2013). Our implementation principle for this section of *PhysLib* is to first create different namespaces and independent Lean files for each topic. Subsequently, we add topic-specific unit types and constants in the topic namespace. Then, we implement basic physics rules summarized from experiments as definitions. Finally, we implement theorems with their proofs that are practical to the topic as the final layer. We implement the mechanics field in detail as an example and set a basic foundation for other topics. We present the design process to the community and launch this project for collaborative development of the field.

2.1.3 COMMUNITY-DRIVEN AND EXTENSIBILITY

As mentioned before, *PhysLib* is designed to be a community-driven and collaborative work like Mathlib Community (2019), and we make our best effort to ensure other researchers can easily read and extend the system while maintaining consistency. In general, we organize *PhysLib* in a three-layer hierarchy: (1) Foundation unit system, which should be consistent with changes only

necessary. (2) Topic-specific unit system, which should be added when formalizing theorem statements where current units are unable to support the construction. (3) Topic-specific theorems, which include most of the practical theorems to support proof implementation and should be regularly updated.

The current version contains a relatively comprehensive implementation of the repository in mechanics, serving as an example for the community. *PhysLib* inherits components from Tao (2025), which are distributed under the Apache-2.0 license, and the same license applies to newly added content. We will strongly encourage contributions of new statements and proofs, and will continuously maintain the repository in the future.

Beyond the current repository we are implementing, such a layered design is applicable to the formalization of other domains, such as the natural and social sciences, and be alternatively extended to general proving systems based on the same logic.

2.2 LEANPHYSBENCH

With the current trend of utilizing LLMs to perform formal reasoning, it is crucial to evaluate their formal physics reasoning capability. However, to the best of our knowledge, there is no dataset for benchmarking the Lean4 physics reasoning capability for LLMs. Thus, we propose (to the best of our knowledge) the first benchmark for Lean4 physics reasoning. In this section, we detail the process of creating this benchmark. We firstly present the data collection process in Section 2.2.1, then detail how we created the benchmark in Section 2.2.2, and report the benchmark statistics in Section 2.2.3.

2.2.1 DATA COLLECTION & PREPROCESSING

Corresponding to the bottom-up principle, when we design the *PhysLib*, when creating the dataset, we follow a basic-to-advanced principle. Our benchmark primarily consists of two levels of data: the high school Olympiad-competition-related level and the college level. The Olympiad data are collected from competition-related exercise books. This level of data focuses on testing the model's capability to perform multi-step reasoning within a specific field of knowledge. On the other hand, the college-level problems are collected from the elementary university physics textbooks. These problems are selected to cover a deeper range of concepts with easier reasoning steps that focus on testing the LLM's reasoning when the topics span multiple physics models. For questions accompanied by figures, we extract the information from the figures and describe it in natural language, alongside the problems. Detailed data sources are provided in Appendix B.

Following *PhysLib*'s design, we further divided the topic of the *LeanPhysBench* into mechanics, waves & acoustics, thermodynamics, electromagnetism, optics, and modern physics. After the data collection, we have the base natural language statement for formulating *LeanPhysBench*.

2.2.2 FORMALIZATION PIPELINE

Following the collection and preprocessing of the NL data, we apply a strict pipeline to transform the NL data into verifiable Lean4 theorems. The overview of the formalization pipeline is shown in Figure 2. Formally, if we denote a physics problem we want to formalize by P, the original problem we have is $P_{original}$, the target is to obtain the Lean4 version of the problem P_{Lean}. The formalization process is as follows:

NL Format Alignment According to previous work in formalizing mathematical statements (Wang et al., 2025b; Zheng et al., 2021), there is a significant representation gap between Lean4 statements and their corresponding NL problems in the original datasets. Specifically, in NL problems, it is typical for the problems to be in question-answering format, where the target is to find a specific numerical or formulaic answer. However, in Lean4, the problem type is a closed-end proof rather than a specific answer, which leads to a gap in the statement. Inspired by Wang et al. (2025b), we perform a format alignment to transform the question-answering style physics problem into a proof statement.

The first step in format alignment is to restate the question-answering problem in a proof format. Specifically, we transform the question part of "Finding the answer to ..." into "Prove that the

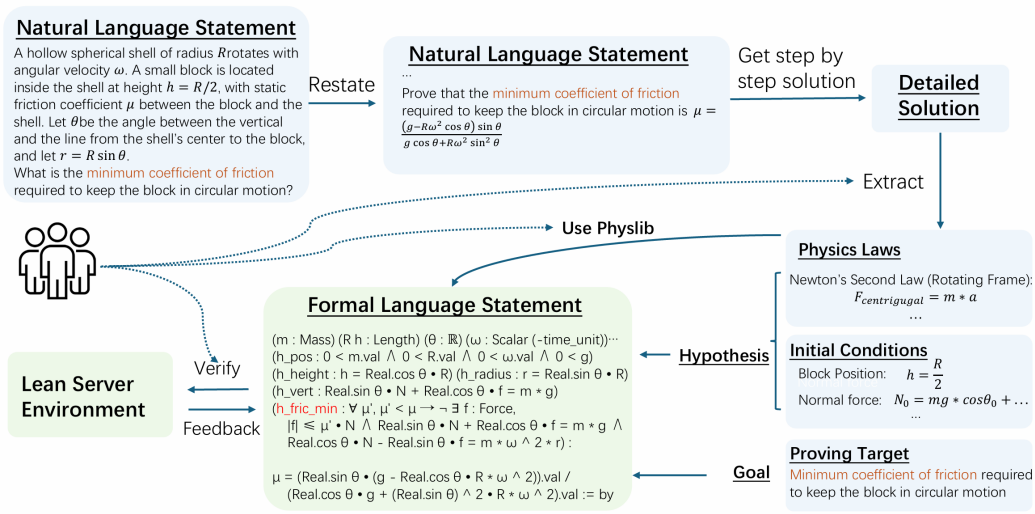

Figure 2: **Overview of *LeanPhysBench* formalization process:** We formalize the *LeanPhysBench* data by the following steps: (1) NL Format Alignment; and (2) Lean4 code writing & verification

College Level

```
theorem University_Mechanics_3
 (x_0 x_1 : Length)
 (t_0 t_1 dt t : Time)
 (v_0 v_1 : Speed)
 (a: Acceleration)
 (xf xf1: Time → Length)(vf : Time → Speed)
 (ht0 : t_0 = 0 · second)
 (ht1 : t_1 = 4 · second)
 (ht : dt = t_1 − t_0)
 (ha : a = (v_1 −v_0)/dt)
 (hv : ∀ t, vf t = v_0 + a * t/1)
 (hv1: v_1 = vf dt)
 (hx: ∀ t, xf t = (a * t**2)/2 + v_0 * t)
 (hxx: ∀ t, xf1 t = (3 · meter / second**2)* t**2 −
  2 · meter / second * t)
 (hxxx: xf = xf1):
 (a = 6 · meter / second**2 ∧
  v_0 = −2 · meter / second)
:= by sorry
```

Competition Level

```
theorem competition_mechanics_Ch2_Q32
 (m : Mass) (R : Length) (θ : ℝ) (v : Speed) (μ : ℝ) (N f : Force)
 (h_pos : 0 < m.val ∧ 0 < μ ∧ 0 < g.val)
 (h_sin_cos : Real.sin θ ≠ 0 ∧ Real.cos θ ≠ 0)
 (r_def : r = Real.sin θ · R)
 (h_horiz : Real.sin θ · N − Real.cos θ · f =
  m * v**2 / r)
 (h_vert : Real.cos θ · N + Real.sin θ · f = m * g)
 (f_def : f = m * (Real.sin θ · g −
  Real.cos θ · v**2 / r / 1))
 (N_def : N = m * (Real.cos θ · g +
  Real.sin θ · v**2 / r / 1))
 (fric_bound : ‖f.val‖ ≤ μ * ‖N.val‖) :
 ∀ (ε : ℤ), (ε = 1 ∨ ε = −1) →(
  f = (ε : ℝ) · μ · N → v**2 =
  ((Real.sin θ − (ε : ℝ) * μ * Real.cos θ) · g *
  (Real.sin θ · R)/
  (Real.cos θ + (ε : ℝ) * μ * Real.sin θ)))
:= by sorry
```

Figure 3: Two examples from *LeanPhysBench* demonstrating different difficulty levels.

answer is ..." following Zheng et al. (2021). Subsequently, to better model the physical process and relations, we require the LLM to write a step-by-step solution to the question. Based on the solution, we extract the physics laws used in the problems and all the initial conditions for composing statements. Finally, we define the proving target for the problem. We split the target of proof into three categories, namely, numerical value, physical expression, or logical formula describing a physical state and use such targets to assist the process of Lean4 code writing. After this step, we align the NL question-answering problem to a provable question with extracted physics laws, initial conditions, and proving target.

Lean4 code writing & Verification After we obtain aligned NL problems, the Lean code writing is split into the formalization of conditions and goals. During the process of writing Lean4 conditions for the problem, we add the necessary physics laws to *PhysLib* as well as properly present such laws in Lean code. In formalizing goals, we write the corresponding Lean4 expression of the proving targets as goals for the Lean4 proof. After we obtain the Lean4 code, it is submitted to the verifier to verify whether it compiles successfully. Additionally, we ask experts for physics and Lean to check the completeness of the theorem. Each statement requires one expert to formalize and at least two experts to verify.

The above pipeline ensures that problems in *LeanPhysBench* accurately capture the underlying physics semantics from NL descriptions. We present two examples of Lean4 statements we for-

malized in Figure 3. In the example, we can clearly see that the college-level problems focus on relatively simple operations in a broader range of topics. Competition-level statements emphasize derivations of formulas using more advanced math tools in a more concentrated field of physics.

2.2.3 BENCHMARK STATISTICS

The detailed statistics for the *PhysLib* are presented in Figure 4. In total, *PhysLib* contains 200 physics statements formalized in Lean4. Among them, 104 statements are at the college-level and 96 are at the competition level. The competition level is further divided into easy (62 problems) and hard (34 problems) categories. Where easy problems focus more on mathematical deduction with looser conditions, and hard problems focus more on physics problem deductions with more physics formulas and tighter conditions.

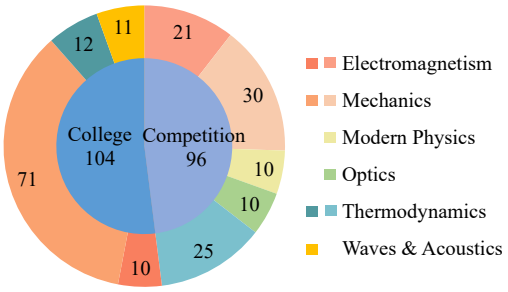

Figure 4: Statics of *LeanPhysBench*: The distribution of 200 Lean4 physics statements across difficulty levels and topics

3 EXPERIMENT

We conduct comprehensive experiments on Lean4PHYS to demonstrate the significance of our proposed *LeanPhysBench* and the effectiveness of *PhysLib*. In particular, we demonstrate the performance of major expert Lean provers and general models on *LeanPhysBench* and prove the effectiveness of *PhysLib* in Section 3.2. We study the problem format of *LeanPhysBench* in Section 3.3, and in Section 3.4 we perform case studies to address important concerns of the main results.

3.1 EXPERIMENT SETUP

We evaluate the LLMs' Lean4 physics reasoning capabilities by applying them to write proofs for the *PhysLib*. Specifically, the task for the LLMs is to compose Lean4 proofs with provided NL statements and Lean4 statements. To ease the load of LLMs in proving, we manually configure all the imports and namespaces. Furthermore, unless otherwise specified, we allow the LLM to use Long Chain-of-Thought capability to perform deeper reasoning. We apply the pass@16 metric to evaluate the performance.

To better demonstrate current LLMs' formal physical reasoning capability, we select the most representative closed-source and open-source models to evaluate. Namely, for closed-source general LLMs, we select GPT-4o (OpenAI et al., 2024), Claude-Sonnet-4 Anthropic (2025), and Gemini-2.5-Pro (Comanici et al., 2025) as baselines. For open-source models, we present both the results of general-purpose models, including DeepSeek-R1-0528 (Guo et al., 2025) and Qwen3-8B (Team, 2025), as well as expert Lean4 provers such as Goedel-Prover-V2-8B (Lin et al., 2025b), Kimina-Prover (Wang et al., 2025a), and DeepSeek-Prover-V2 (Ren et al., 2025).

Furthermore, to demonstrate the effectiveness of *PhysLib*, we test the LLM's capability under modes that have or do not have *PhysLib* in the generated context. In summary, we test the *LeanPhysBench* on eight major LLMs in two modes. The implementation details of our experiments can be found in Appendix C.

3.2 MAIN RESULT

We demonstrate our main experiment result in Table 1. The result demonstrates that LLMs that are larger in size and have better coding capabilities, like Gemini and Claude, perform better in formal physics reasoning. With *PhysLib* in context, Gemini achieves a comparatively higher accuracy rate of 40.50% for the entire dataset, while Claude obtains 35.00%. In comparison, the expert provers and GPT-4o suffer from suboptimal performance with an overall accuracy rate of around 10%. Furthermore, the results demonstrate that with *PhysLib* added to the context, performance improved by **11.88%** and such improvement is consistent across all models and difficulty levels. It indicates the effectiveness of our *PhysLib* in assisting LLMs' formal physics reasoning.

Table 1: Pass@16 results of *LeanPhysBench* on 8 LLMs in with (✓) and without (✗) *PhysLib* mode across different difficulty levels, including College, Competition-Easy (Comp-Easy), and Comp-Hard. The best result is **bolded** and the second-best result is underlined.

Method	PhysLib	College	Comp-Easy	Comp-Hard	Overall
Open-source models					
DeepSeek-R1-8B (Guo et al., 2025)	✓	6.73%	9.68%	0.00%	6.50%
	✗	0.00%	4.84%	0.00%	1.50%
Qwen3-8B (Team, 2025)	✓	7.69%	8.06%	2.94%	7.00%
	✗	1.92%	3.23%	0.00%	2.00%
Kimina-Prover-8B (Wang et al., 2025a)	✓	8.65%	20.97%	**11.76%**	13.00%
	✗	5.77%	16.13%	**11.76%**	10.00%
Goedel-Prover-V2-8B (Lin et al., 2025a)	✓	8.65%	24.19%	**11.76%**	14.00%
	✗	6.73%	19.35%	8.82%	11.00%
DeepSeek-Prover-V2-7B (Ren et al., 2025)	✓	9.62%	29.03%	**11.76%**	16.00%
	✗	6.73%	22.58%	**11.76%**	12.50%
Closed-source models					
GPT-4o (OpenAI et al., 2024)	✓	6.73%	30.65%	2.94%	13.50%
	✗	2.88%	1.61%	2.94%	2.50%
Claude-Sonnet-4 (Anthropic, 2025)	✓	28.85%	62.90%	2.94%	35.00%
	✗	0.96%	4.84%	2.94%	2.50%
Gemini-2.5-pro (Comanici et al., 2025)	✓	**31.73%**	**74.19%**	5.88%	**40.50%**
	✗	6.73%	12.90%	5.88%	8.50%

Upon closer examination, we can observe that Lean experts, which significantly outperform closed-source general LLMs in the math domain, lack the strong formal physics capabilities of these models. This reveals that the expert Lean provers' capabilities are limited to the math domain and fail to transfer to other general domains, especially when these domains apply a new definition (such as the unit system in Lean4PHYS). Moreover, we found that the performance difference between expert provers is relatively marginal, indicating that significant improvement in mathematics does not guarantee a large improvement in physics reasoning.

When analyzing the results from different levels of difficulty, we find that the models generally perform well on easy problems in competition problems, which are more closely aligned with mathematical representations. However, expert models still do not outperform larger closed-source models, indicating the limited transfer of capability from math to physics formal reasoning in Lean4. Such a finding is also true for college-level problems, where the performance of models is generally lower. However, for the competition-hard level of problems, the expert provers perform better than the closed-source models. This is because the expert models have a stronger capability to perform complex deduction, whereas large models do not.

Furthermore, adding *PhysLib* to the context will significantly improve the LLMs' performance. This is because when adding the *PhysLib*, the model has a better understanding of the formulation of the basic system we use to compose the proof. Without the *PhysLib*, the model can only perform basic simplification tactics like: `constructor`, `rw`, `abel`, `exact`, `aesop`. By adding the *PhysLib*, the model can learn to perform more advanced tactics, such as `simp` and `norm_num`, based on the theorems and definitions in *PhysLib*.

3.3 Problem Format Study

This section presents a more detailed study of the problem format. Due to the space limit, we place the example in Figure 5 in the Appendix and only provide the analysis result here.

From the example of problem format, we observe that college-level statements primarily involve numeric computations with a relatively wide range of physical quantities with units. These problems rarely require multi-step formula derivations, but heavily depend on the unit system in *PhysLib* to ensure dimensional consistency. Thus, the models with weaker in-context learning perform rel-

atively badly on this level of problems. It is because they cannot infer the new out-of-distribution syntax or unit-handling rules from context.

The easy-level competition questions are closer to transitional math problems in Lean, such as MiniF2F (Zheng et al., 2021). They involve relatively simple formula derivations, typically within two steps and often only tactics from Mathlib. Therefore, the models that are more familiar with the Lean mathematics and good at using tactics perform better than closed-source large models on these kinds of problems even without *PhysLib*.

On the other hand, the hard level of competition problems demands complex symbolic reasoning, handling quantifiers, and difficult functional reasoning. For instance, this includes proving the existence of a number, holding of an inequality, or deriving a functional relation. These problems combine unit casts with symbolic manipulation, further increasing the difficulty. Solving this level of problem requires careful decomposition of the problem, proving new lemmas, and diverse proof strategies with creativity. Moreover, many problems of this level require calculus concepts such as limits and continuity. All the above factors combined cause the low pass rate at this level.

3.4 Case Study

We present the case analysis in this section to provide a more detailed examination of the key findings from our main experiment. Due to the space limit, for a detailed example, please refer to Appendix E.2.

Behavior of the same theorem with and without *PhysLib* Figure 6 demonstrates a theorem at the college level, which Gemini can do both with and without *PhysLib* in context. From the comparison, we find that without the library, the proof is based solely on Mathlib. When the *PhysLib* is in the context, the proof tends to use the operations in the library and include explanatory comments. It indicates *PhysLib* can assist LLM's reasoning by providing a wider toolbox.

Transfer tricks in mathematics Figure 9 demonstrate a problem only solved by Goedel-Prover-V2-8B in our entire cycle of experiment. We conclude that this is a good transfer of following the NL intermediate steps and performing multiple trials learned in the math Lean training. From the NL statement demonstrated in Figure 8, we can observe that most steps are presented in the statement. Following these NL hints and many trials using `try` tactics, the Goedel-Provers can successfully solve the problem. It indicates that, although limited, the expert prover's capability of Lean math reasoning can be applied to formal physics reasoning.

Why the general model performs better To answer this question, we further analyze different topics solved by LLMs and find that general LLMs are typically better in thermo-dynamics. The Gemini successfully proved 22 theorems in the field, but DeepSeek-Prover-V2 only finished six theorems. We present one theorem that is proved by both Gemini-2.5 and DeepSeek-Prover-V2 to study the different behavior of their proof in Figure 6. We observe that the proof from the expert prover is more complex and tedious, while Gemini's proof is cleaner and more straightforward. Such a difference in proving consistently shows between the expert prover and general models. It suggests that in fields underrepresented in training, the prolonged deliberation of expert models may lead to overthinking and result in suboptimal results.

4 Related Work

4.1 Formal Reasoning

Formal languages (FL) are introduced in the mathematical domain to ensure that proofs are logically correct, verifiable, and free from errors that human reasoning might overlook. Lean de Moura & Ullrich (2021) is such a formal proof language and interactive theorem prover, based on dependent type theory and supported by the large-scale Mathlib library Community (2019). With the introduction of large language models (LLMs) to assist in generating such proofs, researchers from the Lean community have made significant progress. Datasets and benchmarks serve as essential foundations for evaluating and advancing these systems. MiniF2F Zheng et al. (2021) introduced cross-system benchmarking by translating competition-level mathematics problems into verifiable

lean statements, while FormalMATH Yu et al. (2025) extended this effort to a scale of over 5,000 problems. LeanDojo Yang et al. (2023) extracted 98,734 theorems and proofs from the Mathlib library.

In parallel, increasing attention is being devoted to improving methods for auto-formalization, translating natural-language statements into machine-checkable proofs. FormL4 Luong et al. (2024) introduced a Process-Supervised Verifier model that leverages the precise feedback from Lean 4 compilers to enhance autoformalization, while TheoremLlama Wang et al. (2024a) adapts general-purpose LLMs into Lean 4 experts through dataset alignment and iterative training. Model-Collaboration strategies Wang et al. (2025c) further enhance proof synthesis by separating the cognition tasks of general NL for whole-proof generation and error analysis for proof correction.

Several Lean-expert LLMs have also been developed to specialize in formal reasoning. DeepSeek-Prover (Ren et al., 2025) advances formal mathematical reasoning via reinforcement learning for subgoal decomposition. Goedel-Prover (Lin et al., 2025a) addresses the scarcity of formalized mathematical statements by training LLMs to translate natural-language mathematics problems into Lean4 statements, and by further training a series of provers alongside an expanding dataset. Kimina-Prover (Wang et al., 2025a) introduces a reasoning-driven exploration paradigm to improve proof search and synthesis.

4.2 Lean in Subjects Beyond Mathematics

Lean has increasingly been explored beyond mathematics, including in chemical physics Bobbin et al. (2023), molecular simulations Ugwuanyi et al. (2025), and electrical engineering blacksph3re (2025). Projects such as PhysLean Tooby-Smith & contributors (2024) aim to formalize mechanics and high-energy physics using new tools like tensor and index notation Tooby-Smith (2024). However, these efforts remain theorem-specific, small-scale, non-modular, and lack standardized benchmarks. While Lean-expert models achieve strong performance on mathematical benchmarks, their ability to transfer formal reasoning skills to other domains, such as physics, remains largely unexamined.

4.3 Physics Datasets

Physics problem datasets for machine learning are widely studied in Natural Language (NL). Several benchmarks are created focusing on physical reasoning and understanding, such as PhysBench Chow et al. (2025) and PHYBench Qiu et al. (2025). At the curriculum level, UGPhysics Xu et al. (2025) and the PHYSICS dataset Zheng et al. (2025) provide thousands of textbook-style problems. At the competition level, more challenging benchmarks appear. There are OlympiadBench He et al. (2024) with 8,476 bilingual Olympiad problems, HiPhO He et al. (2024) with recent International Physics Olympiad questions, and PhysReason Zhang et al. (2025), which emphasizes multi-step reasoning and diagram understanding. Also, the CAMEL-Physics Lu et al. (2025) scales to tens of thousands of automatically generated problems. These resources are valuable for assessing LLM reasoning skills in physics in NL.

5 Conclusion

This paper presents Lean4PHYS, a comprehensive framework to support Lean4 physics reasoning. The framework includes *PhysLib*, an extensible, community-driven foundation library that sets the cornerstone for units, fields, and theorems for formal physics reasoning. To evaluate LLMs' performance on formal physics reasoning, we propose *LeanPhysBench*, (to the best of our knowledge)the first benchmark for Lean4 physics reasoning. Based on the Lean4PHYS, we conduct extensive experiments to provide an overview of LLMs' performance on such tasks and the effectiveness of our *PhysLib*. We find that expert provers do not outperform large general models in most cases of formal physics reasoning. It indicates limited transfer capability from mathematical reasoning, despite the fact that they are all Lean4-based. Furthermore, the experiment shows that with *PhysLib* in the context, LLMs' performance on *LeanPhysBench* increases by **11.88%** on average. Beyond formal physics reasoning, our work provides a general principle for formalizing natural science beyond mathematics into a verifiable system.

Ethics Statement

This work focuses on formalizing college-level physics problems in Lean4 and evaluating LLMs' formal physics reasoning capabilities. To the best of our knowledge, we carefully followed the ethical regulations of the conference. We adhere to the following ethical considerations:

- **Data Sources:** All physics problems are sourced from publicly available university textbooks and competition exercise books. Proper citations are provided, and no private or sensitive data was used.
- **Responsible Use of LLMs:** LLMs were only used for polishing the writing style and grammar error detaction.
- **Licensing and Attribution:** The parts of the *PhysLib* library from Tao (2025) are licensed with Apache-2.0 license, aligned with the origin repository.

Reproducibility Statement

We have made efforts to ensure that our work is reproducible. The detailed description of the Lean4PHYS framework is presented in Section 2 with experiment details provided in Section 3 and Appendix C, D. We plan to open source the code of Lean4PHYS, consisting of a community-driven repository and a benchmark in the near future.

References

Anthropic. Claude sonnet 4, 2025. URL https://www.anthropic.com/claude/sonnet. Accessed: 2025-09-23.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023.

blacksph3re. distribution_factors: An attempt to formalize dc loadflow equations and distribution factors in lean, 2025. URL https://github.com/blacksph3re/distribution_factors. Accessed: 2025-09-25.

Maxwell P. Bobbin, Samiha Sharlin, Parivash Feyzishendi, An Hong Dang, Catherine M. Wraback, and Tyler R. Josephson. Formalizing chemical physics using the lean theorem prover. *Digital Discovery*, 3(2):264–280, 2023. doi: 10.1039/D3DD00077J. URL https://pubs.rsc.org/en/content/articlelanding/2024/dd/d3dd00077j.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Guizilini, and Yue Wang. Physbench: Benchmarking and enhancing vision-language models for physical world understanding, 2025. URL https://arxiv.org/abs/2501.16411.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

The Lean Mathematical Library Community. The lean mathematical library. *arXiv*, abs/1910.09336, 2019. URL https://arxiv.org/abs/1910.09336. Accessed: 2025-09-23.

Projet Coq. The coq proof assistant-reference manual. *INRIA Rocquencourt and ENS Lyon, version*, 5, 1996.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction - CADE-28: 28th International Conference on Automated Deduction*, pp. 625–635, Berlin, Heidelberg, 2021. Springer-Verlag. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pp. 378–388. Springer, 2015.

Kefan Dong and Tengyu Ma. Stp: Self-play llm theorem provers with iterative conjecturing and proving, 2025. URL https://arxiv.org/abs/2502.00212.

Carl Edwards, Chi Han, Gawon Lee, Thao Nguyen, Bowen Jin, Chetan Kumar Prasad, Sara Szymkuć, Bartosz A Grzybowski, Ying Diao, Jiawei Han, et al. mclm: A function-infused and synthesis-friendly modular chemical language model. *arXiv preprint arXiv:2505.12565*, 2025.

Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno de Moraes Dumont, and Sanmi Koyejo. Putnam-axiom: A functional and static benchmark for measuring higher level mathematical reasoning. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals of physics*. John Wiley & Sons, 2013.

John Harrison. Hol light: An overview. In *International Conference on Theorem Proving in Higher Order Logics*, pp. 60–66. Springer, 2009.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-prover: A frontier model for open-source automated theorem proving, 2025a. URL https://arxiv.org/abs/2502.07640.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, et al. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025b.

Dakuan Lu, Xiaoyu Tan, Rui Xu, Tianchu Yao, Chao Qu, Wei Chu, Yinghui Xu, and Yuan Qi. Scp-116k: A high-quality problem-solution dataset and a generalized pipeline for automated extraction in the higher education science domain, 2025. URL https://arxiv.org/abs/2501.15587.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*, 2024. URL https://arxiv.org/abs/2406.01940.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pp. 625–635. Springer, 2021.

David B Newell, Eite Tiesinga, et al. The international system of units (si). *NIST Special Publication*, 330(1), 2019.

Aaron Hurst OpenAI, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 1(2):3, 2024.

Lawrence C Paulson. *Isabelle: A generic theorem prover*. Springer, 1994.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, Chenyang Wang, Chencheng Tang, Haoling Chang, Qi Liu, Ziheng Zhou, Tianyu Zhang, Jingtian Zhang, Zhangyi Liu, Minghao Li, Yuku Zhang, Boxuan Jing, Xianqi Yin, Yutong Ren, Zizhuo Fu, Jiaming Ji, Weike Wang, Xudong Tian, Anqi Lv, Laifu Man, Jianxiang Li, Feiyu Tao, Qihua Sun, Zhou Liang, Yushu Mu, Zhongxuan Li, Jing-Jun Zhang, Shutao Zhang, Xiaotian Li, Xingqi Xia, Jiawei Lin, Zheyu Shen, Jiahang Chen, Qiuhao Xiong, Binran Wang, Fengyuan Wang, Ziyang Ni, Bohan Zhang, Fan Cui, Changkun Shao, Qing-Hong Cao, Ming xing Luo, Yaodong Yang, Muhan Zhang, and Hua Xing Zhu. Phybench: Holistic evaluation of physical perception and reasoning in large language models, 2025. URL https://arxiv.org/abs/2504.16074.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. Scienceqa: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23(3):289–301, 2022.

Yousheng Shu, Wangyu Hu, and Bingqian Chen. *Selected Advanced Physics Problems, Volume II*. Science Press, Beijing, 2008. ISBN 9787030193563.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Terence Tao. A lean companion to analysis i. https://github.com/teorth/analysis, 2025. URL https://github.com/teorth/analysis. GitHub repository, accessed: 2025-08-31.

Qwen Team. Qwen3, April 2025. URL https://qwenlm.github.io/blog/qwen3/.

Joseph Tooby-Smith. Formalization of physics index notation in lean 4. *arXiv preprint arXiv:2411.07667*, 2024. URL https://arxiv.org/abs/2411.07667. Integrating tensor index notation into Lean 4 for formal verification.

Joseph Tooby-Smith and contributors. Physlean: Digitalising physics in lean 4. *arXiv preprint arXiv:2405.08863*, 2024. URL https://arxiv.org/abs/2405.08863. An open-source project to digitalise results from physics in Lean 4.

Ejike D. Ugwuanyi, Colin T. Jones, John Velkey, and Tyler R. Josephson. Benchmarking energy calculations using formal proofs. *arXiv preprint arXiv:2505.09095*, 2025. URL https://arxiv.org/abs/2505.09095.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025a.

Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. Theorem-llama: Transforming general-purpose llms into lean4 experts. *arXiv preprint arXiv:2407.03203*, 2024a.

Ruida Wang, Yuxin Li, Yi R Fung, and Tong Zhang. Let's reason formally: Natural-formal hybrid reasoning enhances llm's math capability. *arXiv preprint arXiv:2505.23703*, 2025b.

Ruida Wang, Rui Pan, Yuxin Li, Jipeng Zhang, Yizhen Jia, Shizhe Diao, Renjie Pi, Junjie Hu, and Tong Zhang. Ma-lot: Model-collaboration lean-based long chain-of-thought reasoning enhances formal theorem proving. *arXiv preprint arXiv:2503.03205*, 2025c.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024b.

Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Kai Shen. Bfs-prover: Scalable best-first tree search for llm-based automatic theorem proving. *arXiv preprint arXiv:2502.03438*, 2025.

Xin Xu, Qiyun Xu, Tong Xiao, Tianhao Chen, Yuchen Yan, Jiaxin Zhang, Shizhe Diao, Can Yang, and Yang Wang. Ugphysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. *arXiv preprint arXiv:2502.00334*, 2025. URL https://arxiv.org/abs/2502.00334.

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, 2023. URL https://arxiv.org/abs/2306.15626.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean workbook: A large-scale lean problem set formalized from natural language math problems. *arXiv preprint arXiv:2406.03847*, 2024.

Hugh D. Young and Roger A. Freedman. *University Physics with Modern Physics, Global Edition*. Pearson Education, 15th edition, 2019. ISBN 9781292314815.

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, Yandong Wen, Ge Zhang, and Weiyang Liu. Formalmath: Benchmarking formal mathematical reasoning of large language models, 2025. URL https://arxiv.org/abs/2505.02735.

Xinyu Zhang, Yuxuan Dong, Yanrui Wu, Jiaxing Huang, Chengyou Jia, Basura Fernando, Mike Zheng Shou, Lingling Zhang, and Jun Liu. Physreason: A comprehensive benchmark towards physics-based reasoning, 2025. URL https://arxiv.org/abs/2502.12054.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

Shenghe Zheng, Qianjia Cheng, Junchi Yao, Mengsong Wu, Haonan He, Ning Ding, Yu Cheng, Shuyue Hu, Lei Bai, Dongzhan Zhou, Ganqu Cui, and Peng Ye. Scaling physical reasoning with the physics dataset. *arXiv preprint arXiv:2506.00022*, 2025. URL https://arxiv.org/abs/2506.00022.

A USE OF LLMS

LLMs were employed to assist in the development and verification of physics problems. Specifically, they served two main roles:

- LLMs generated detailed, step-by-step solutions for physics problems to help annotators better understand the underlying reasoning and calculations.
- LLMs checked the correctness and completeness of natural-language solutions. All outputs were subsequently reviewed and verified by human annotators to ensure accuracy and reliability.

LLMs were also used to improve the clarity and readability of the manuscript. Tasks included:

- Correcting grammar and sentence structure.
- Formatting figure captions and table layouts consistently.

B DATA SOURCES

Textbook-level questions are adapted from the concepts presented in a university textbook Young & Freedman (2019) and UGPhysics Xu et al. (2025). Olympiad-Easy questions are derived from intermediate steps of competition problems, while Olympiad-Hard questions are based on ideas from a physics Olympiad practice book Shu et al. (2008). UGPhysics is distributed under the CC BY-NC-SA 4.0 license. For the textbooks published by Pearson Young & Freedman (2019) and Science Press Shu et al. (2008), we strictly respect their copyright: rather than copying the questions verbatim, we reformulated and rephrased them based on the underlying physics ideas.

C IMPLEMENTATION DETAIL

The generation configuration for the LLM roll-out of the experiment is as follows:

- Top-p: 0.95
- Temperature: 0.8
- Maximum tokens per generation: 16,384
- Repetition penalty: 1.0

Open-source models are tested under a 4-card H20 server. The entire open-source LLM roll-out process costs about 2 days.

D LLM PROMPT TEMPLATES

We provide the prompt templates used to guide LLMs in generating Lean4 proofs.

With *PhysLib*:

```
Please first learn the new library besides mathlib and usage
↪  examples before answering the question. You should refer to
↪  the new unit system.

{PhysLib}

Complete the following Lean 4 code.
Provide your response in two parts, each enclosed in separate
↪  markdown code blocks:

```plan
# Proof Plan
- Outline the main proof steps and strategies.
- Highlight key intermediate lemmas and structures.
- Describe how to connect them to form the final proof.
```
```lean4
{Lean4_header}

/-- {NL_statement} -/
{Lean4_statement}
```
```

Without *PhysLib*:

```
Complete the following Lean 4 code.
Provide your response in two parts, each enclosed in separate
↪  markdown code blocks:

```plan
# Proof Plan
- Outline the main proof steps and strategies.
- Highlight key intermediate lemmas and structures.
- Describe how to connect them to form the final proof.
```
```lean4
{Lean4_header}

/-- {NL_statement} -/
{Lean4_statement}
```
```

E EXPERIMENT

E.1 FORMAT STUDY

The differences between the questions are shown in Figure 5. W can see that college-level statements primarily involve numeric computations with a relatively wide range of physical quantities with units and without multi-step formula derivations. The easy-level competition questions are closer to transitional math problems in Lean and involve relatively simple formula derivations, typically within two steps and often only tactics from Mathlib. The hard level of competition problems demands complex symbolic reasoning, handling quantifiers, and difficult functional reasoning , in-

cluding proving the existence of a number, holding of an inequality, or deriving a functional relation. Advanced mathematics concepts such as derivatives, integrals, limits and continuity may be involved.

E.2 Case Study

Behavior of the same theorem with and without *PhysLib* Figure 6 illustrates a college-level theorem that Gemini can prove both with and without *PhysLib* in the context. The comparison shows that, without the library, the proof relies entirely on Mathlib. When *PhysLib* is available, the proof incorporates operations from the library and includes explanatory comments. This suggests that *PhysLib* can enhance LLM reasoning by offering a broader set of tools.

Transfer tricks in mathematics Figure 9 presents a problem that was solved only by Goedel-Prover-V2-8B across our entire experimental cycle. This demonstrates effective transfer of skills learned from Lean mathematics training, specifically following natural-language intermediate steps and performing multiple trial attempts. From the NL statement shown in Figure 8, we can see that most proof steps are explicitly provided. By leveraging these NL hints and repeatedly applying `try` tactics, the Goedel-Prover successfully solves the problem. This suggests that, although limited, the reasoning capabilities of Lean math expert models can be extended to formal physics tasks.

Why the general model performs better To investigate this question, we further analyze the performance of LLMs across different topics and find that general LLMs tend to perform better in thermodynamics. For example, Gemini successfully proved 22 theorems in this field, whereas DeepSeek-Prover-V2 completed only six. Figure 6 illustrates one theorem proved by both Gemini-2.5 and DeepSeek-Prover-V2, highlighting differences in their proof strategies. We observe that the expert prover's proof is more complex and tedious, while Gemini's proof is cleaner and more straightforward. This pattern of differences consistently appears between expert provers and general models, suggesting that in fields underrepresented in training, the prolonged deliberation of expert models may lead to overthinking and suboptimal results.

College

```
Two point charges are located on the x-axis
     of a coordinate system: q1 = 1.0 nC is
     at x = +2.0 cm, and q2 = -3.0 nC is at
     x = +4.0 cm. What is the total
     electric force exerted by q1 and q2 on
     a charge q3 = 5.0 nC at x = 0?
theorem Electromagnetism_3_University
    (q1 q2 q3 : Charge) (x1 x2 x3 : Length)
    (hq1 : q1 = SI.nano (1  · coulomb))
    (hq2 : q2 = SI.nano (-3 · coulomb))
    (hq3 : q3 = SI.nano (5  · coulomb))
    (hx1 : x1 = ((0.02:ℝ) · meter))
    (hx2 : x2 = ((0.04:ℝ) · meter))
    (hx3 : x3 = 0)(F  : Force)
    (hF : F = K * q3 * (q1 / (x1 - x3)^2 +
     q2 / (x2 - x3)^2)):
    F = ((9:ℝ)*(10:ℝ)^(-22:ℚ)/32) · newton
     := by
  simp [←Scalar.val_inj, hF, hq1, hq2,
     hq3, hx1, hx2, hx3, K, SI.nano,
     coulomb, meter, newton]
  norm_num
```

Competition-Easy

```
The plates of a parallel-plate capacitor
     are 2.50 mm apart, and each carries a
     charge of magnitude 80.0 nC. The
     plates are in vacuum. The electric
     field between the plates has a
     magnitude of \(4.00 \times
     10^{6}\,\text{V/m}\). What is the
     capacitance?
theorem Ch13_electro_question_8
    (V:Voltage)(C:Scalar
     capitance_unit)(q:Charge)
    (E:Scalar
     (force_unit-charge_unit))(d:Length)
    (h: capitance_unit=charge_unit -
     voltage_unit)
    (hC:C=(q/V).cast h)(hq:q=SI.nano (80 ·
     coulomb))
    (hV:V=E*d)(hE:E=(4e6:ℝ) · StandardUnit
     _)
    (hd:d=SI.milli ((2.5:ℝ)· meter)):
    C=(1 / 125000000000:ℚ) · StandardUnit _
     := by
  have hC_expanded : C = (q/(E*d)).cast h :=
      by rw [hC, hV]
  rw [hC_expanded, hq, hE, hd]
  simp [nano, milli, coulomb, meter, ←
     Scalar.val_inj]
  norm_num
```

Competition-Hard

```
Capstan law: If a rope of coefficient of
     friction μ wraps n turns (θ_total =
     2πn) around a post, the tension ratio
     between the heavy side M and the light
     side m satisfies n = (1 / (2πμ)) *
     log(M / m) assuming M > m > 0 and μ >
     0.

theorem Ch2_Q1
  (M m : Mass)
  (μ : ℝ)
  (n : ℝ)
  (θ_total : ℝ := 2 * Real.pi * n)
  (T : ℝ → Force)
  (h_pos : 0 < M.val ∧ 0 < m.val ∧ 0 < μ)
  (hM_gt_m : M.val > m.val)
  (T_light_def : T 0 = m * g)
  (T_heavy_def : T θ_total = M * g)
  (capstan_differential : ∀ θ : ℝ, deriv
     (fun θ' => (T θ').val) θ = μ * (T θ
     ).val)
  (capstan_integral : Real.log ((T θ
     _total).val / (T 0).val) = μ * θ_total)
  (theta_def : θ_total = 2 * Real.pi * n) :
  n = (1 / (2 * Real.pi * μ)) * Real.log
     (M.val / m.val) := by
  rcases h_pos with ⟨hM, hm, hmu⟩
  have h1 : Real.log ((T θ_total).val / (T
     0).val) =
           Real.log ((M * g).val / (m *
     g).val) := by rw [T_heavy_def,
     T_light_def]
  have h2 : Real.log ((M * g).val / (m *
     g).val) = Real.log (M.val / m.val) :=
     by
    have h3 : (M * g).val / (m * g).val =
     M.val / m.val := by
      field_simp; ring_nf; simp; ring
    rw [h3]
  have h3 : Real.log (M.val / m.val) = μ * θ
     _total := by linarith
     [capstan_integral, h1, h2]
  have h4 : Real.log (M.val / m.val) = μ *
     (2 * Real.pi * n) := by rw [theta_def]
     at h3; linarith
  have h5 : μ ≠ 0 := by linarith
  have h6 : Real.pi ≠ 0 := Real.pi_ne_zero
  have h7 : Real.log (M.val / m.val) = 2 *
     Real.pi * μ * n := by linarith [h4]
  have h8 : n = (Real.log (M.val / m.val))
     / (2 * Real.pi * μ) := by field_simp;
     linarith
  rw [h8]; field_simp; ring_nf; field_simp;
     ring
```

Figure 5: Three sampled physics questions from *College Textbook*, *Olympics-Easy*, *Olympics-Hard* problems. Each example shows the natural language problem statement followed by its corresponding Lean formalization with a verified proof.

Gemini-2.5-pro without *PhysLib*

```
theorem Mechanics_73_University
  (T : Force)(m_1 m_2 : Mass)(a : Acceleration)
  (ha : a = m_2 * g / (m_1 + m_2))
  (hT : T = (m_1 * m_2) / (m_1 + m_2) * g):
  (a = (m_2 / (m_1 + m_2)) * g ∧ T = (m_1 * m_2) / (m_1 + m_2) * g) :=
    by
  constructor
  · rw [ha]
    ext
    simp
    rw [mul_div_right_comm]
  · exact hT
```

Gemini-2.5-pro with *PhysLib*

```
theorem Mechanics_73_University
  (T : Force)(m_1 m_2 : Mass)(a : Acceleration)
  (ha : a = m_2 * g / (m_1 + m_2))
  (hT : T = (m_1 * m_2) / (m_1 + m_2) * g):
  (a = (m_2 / (m_1 + m_2)) * g ∧ T = (m_1 * m_2) / (m_1 + m_2) * g) :=
    by
  constructor
  . -- The first goal is to prove the rearranged expression for
    acceleration `a`.
    -- We start by substituting `a` with its given definition from
    hypothesis `ha`.
    rw [ha]
    -- The goal is now to prove an algebraic equivalence between two
    `Scalar` expressions.
    -- We use the `Scalar.val_inj` lemma to transform this into an
    equality of their
    -- underlying real values. This is a standard technique in the
    PhysLib library.
    simp only [← Scalar.val_inj]
    -- Now, with the goal being an equality of real numbers, we can
    use `simp` to apply
    -- the library's rules for how `.val` distributes over arithmetic
    operations.
    simp
    -- The resulting goal is `m_2.val * g.val / (m_1.val + m_2.val) =
    m_2.val / (m_1.val + m_2.val) * g.val`.
    -- This is a standard algebraic identity which the `ring` tactic
    can solve.
    ring
  . -- The second goal is to prove the expression for tension `T`,
    which is identical
    -- to the hypothesis `hT`.
    exact hT
```

Figure 6: Two examples from *LeanPhysBench* demonstrating different proofs of the same college-level problem, generated by Gemini-2.5-pro with and without PhysLib.

Gemini-2.5-pro with *PhysLib*

```
theorem Mechanics_37_University
  (x_1 y_1 d_1 : Length)(t_1 t : Time)(xf1 yf1: Time → Length)
  (ht1 : t_1 = 2 · second)
  (hxx: ∀ t, xf1 t = 2 · meter - ((0.25 : ℚ) · meter / second**2)*
    t**2)
  (hx: x_1 = xf1 t_1)
  (hyy: ∀ t, yf1 t = 1 · meter / second * t + ((0.025 : ℚ) · meter /
    second**3)* t**3)
  (hy: y_1 = yf1 t_1)
  (hd: d_1**2 = (x_1**2 + y_1**2)):
  (x_1 = 1 · meter ∧ y_1 = (2.2 : ℚ) · meter ∧ d_1**2 = (5.84 : ℚ) ·
    meter**2) := by
  -- First, calculate the value of x_1 at t_1.
  have hx_val : x_1 = 1 · meter := by
    -- Substitute the given expressions and values, convert to an
    equality of real numbers,
    -- simplify the units, and compute the final numerical value.
    simp [hx, hxx, ht1, ←Scalar.val_inj, meter, second]
    norm_num

  -- Second, calculate the value of y_1 at t_1.
  have hy_val : y_1 = (2.2 : ℚ) · meter := by
    -- The strategy is the same as for x_1.
    simp [hy, hyy, ht1, ←Scalar.val_inj, meter, second]
    norm_num

  -- Third, calculate the squared distance d_1**2 using the results
    for x_1 and y_1.
  have hd_val : d_1**2 = (5.84 : ℚ) · meter**2 := by
    -- Substitute the formula for d_1**2 and the proven values of x_1
    and y_1.
    simp [hd, hx_val, hy_val, ←Scalar.val_inj, meter]
    norm_num

  -- Finally, use the proven facts to establish the conjunction.
  simp [hx_val, hy_val, hd_val]
```

Figure 7: An example of a problem solved by adding *PhysLib*.

Goedel-Prover with *PhysLib* (nature statement)

Question 74: You want to move a 500 N crate across a level floor. To start the crate moving, you have to pull with a 230 N horizontal force. Once the crate starts to move, you can keep it moving at constant velocity with only 200 N. What are the coefficients of static and kinetic friction? Follow the process: Just before the crate starts to move, we have:

$$\sum F_x = T + \big(-(f_s)_{\text{max}}\big) = 0 \quad \text{so} \quad (f_s)_{\text{max}} = T = 230\ \text{N}$$

$$\sum F_y = n + (-w) = 0 \quad \text{so} \quad n = w = 500\ \text{N}$$

Now we solve $(f_s)_{\text{max}} = \mu_s n$, for the value of μ_s:

$$\mu_s = \frac{(f_s)_{\text{max}}}{n} = \frac{230\ \text{N}}{500\ \text{N}} = 0.46$$

After the crate starts to move, we have:

$$\sum F_x = T + \big(-f_k\big) = 0 \quad \text{so} \quad f_k = T = 200\ \text{N}$$

$$\sum F_y = n + (-w) = 0 \quad \text{so} \quad n = w = 500\ \text{N}$$

Using $f_k = \mu_k n$, we find:

$$\mu_k = \frac{f_k}{n} = \frac{200\ \text{N}}{500\ \text{N}} = 0.40$$

show that the coefficients of static is $\mu_s = 0.4$ and kinetic friction is $\mu_k = 0.40$.

Figure 8: Natural Language statement of a problem solved only by Goedel-Prover.

Goedel-Prover with *PhysLib* (Lean4 statement)

```
theorem Mechanics_74_University_0
  (f_s_max f_k n w : Force)(μ_s μ_k : ℚ)
  (hw : w = 500 · newton)
  (hn : n = w)
  (hf_s_max : f_s_max = 230 · newton)
  (hf_k : f_k = 200 · newton)
  (hμ_s : μ_s = Scalar.val f_s_max / Scalar.val n)
  (hμ_k : μ_k = Scalar.val f_k / Scalar.val n):
  (μ_s = (0.46 : ℚ) ∧ μ_k = (0.40 : ℚ)) := by
  have h1 : Scalar.val f_s_max = 230 := by
    rw [hf_s_max]
    simp [Scalar.val_smul, newton]

  have h2 : Scalar.val n = 500 := by
    have h2₁ : n = 500 · newton := by
      rw [hn, hw]
    rw [h2₁]
    simp [Scalar.val_smul, newton]

  have h3 : (μ_s : ℝ) = (0.46 : ℝ) := by
    have h3₁ : (μ_s : ℝ) = Scalar.val f_s_max / Scalar.val n := by
      norm_cast at hμ_s ⊢
    rw [h3₁]
    rw [h1, h2]
    norm_num

  have h4 : μ_s = (0.46 : ℚ) := by
    have h4₁ : (μ_s : ℝ) = (0.46 : ℝ) := h3
    norm_num at h4₁ ⊢
    norm_num
    (try field_simp at *)
    (try norm_cast at *)

  have h5 : Scalar.val f_k = 200 := by
    rw [hf_k]
    simp [Scalar.val_smul, newton]

  have h6 : (μ_k : ℝ) = (0.40 : ℝ) := by
    have h6₁ : (μ_k : ℝ) = Scalar.val f_k / Scalar.val n := by
      norm_cast at hμ_k ⊢
    rw [h6₁]
    rw [h5, h2]
    norm_num

  have h7 : μ_k = (0.40 : ℚ) := by
    have h7₁ : (μ_k : ℝ) = (0.40 : ℝ) := h6
    norm_num at h7₁ ⊢
    (try field_simp at *)
    (try norm_cast at *)

  have h8 : (μ_s = (0.46 : ℚ) ∧ μ_k = (0.40 : ℚ)) := by
    exact ⟨h4, h7⟩

  exact h8

end Mechanics
```

Figure 9: A problem solved only by Goedel-Prover.

DeepSeek-Prover-V2

```
theorem Ch10_question_4
    (P1 P2:Pressure) (V1 V2:Volume) (T1 T2:Temperature) (k:ℝ)
    (hV1:V1= 13· StandardUnit _)
    (hT1:T1=((1.8*15+273.15):ℝ) · kelvin)
    (hT2:T2=((-44.5*1.8+273.15):ℝ) · kelvin)
    (hP1:P1=((1.01*10^5):ℝ) · pascal)
    (hP2:P2= 868 · pascal)
    (hV2:V2=V1*T2*P1/(T1*P2))
    (hk:k=Real.rpow (V2/V1).val ((1/3):ℝ)):
    k=Real.rpow (10832250 / 144739) (1 / 3) := by
  have h_main : k = Real.rpow (10832250 / 144739) (1 / 3) := by
    have h₁ : k = Real.rpow (V2 / V1).val ((1 / 3 : ℝ)) := by rw [hk]
    rw [h₁]
    have h₂ : (V2 / V1).val = (10832250 / 144739 : ℝ) := by
      -- Simplify the expression for V2 / V1
      simp_all [Pressure, Volume, Temperature, add_assoc, add_comm,
    add_left_comm, mul_assoc, mul_comm, mul_left_comm,
    div_eq_mul_inv, mul_inv_cancel_left]
      <;> ring_nf at *
      <;> norm_num at *
      <;> field_simp [StandardUnit, mul_assoc, mul_comm,
    mul_left_comm] at *
      <;> ring_nf at *
      <;> norm_num at *
      <;> nlinarith
    rw [h₂]
    <;> simp [Real.rpow_def_of_pos]
    <;> norm_num
    <;> ring_nf
    <;> norm_num
    <;> field_simp
    <;> ring_nf
    <;> norm_num
    <;> nlinarith
  exact h_main
```

Gemini-2.5-pro

```
theorem Ch10_question_4
    (P1 P2:Pressure)(V1 V2:Volume)(T1 T2:Temperature)(k:ℝ)
    (hV1:V1= 13· StandardUnit _)
    (hT1:T1=((1.8*15+273.15):ℝ) · kelvin)
    (hT2:T2=((-44.5*1.8+273.15):ℝ) · kelvin)
    (hP1:P1=((1.01*10^5):ℝ) · pascal)
    (hP2:P2= 868 · pascal)
    (hV2:V2=V1*T2*P1/(T1*P2))
    (hk:k=Real.rpow (V2/V1).val ((1/3):ℝ)):
    k=Real.rpow (10832250 / 144739) (1 / 3) := by
  -- Use all hypotheses to substitute the variables and simplify the
    expression.
  -- `simp` will use `hk` to substitute `k`, `hV2` to substitute `V2`,
  -- and the other hypotheses to substitute the numerical values.
  -- The `@[simp]` lemmas in the UnitsSystem library will handle the
    `.val` operations.
  simp [hk, hV2, hV1, hT1, hT2, hP1, hP2]
  -- The goal is now a purely numerical equality. Use `norm_num` to
    verify it.
  norm_num
```

Figure 10: Demonstrating different proof styles on the same problem.