PhysProver: Advancing Automatic Theorem Proving for Physics

Anonymous ACL submission

Abstract

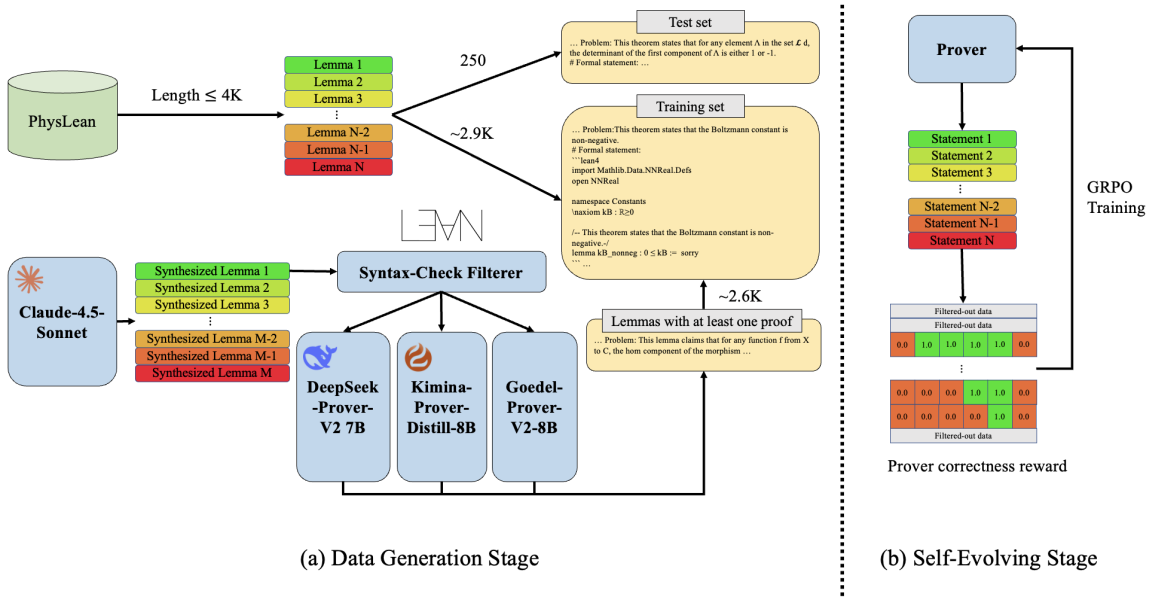
The combination of verifiable languages and LLMs has significantly influenced both the mathematical and computer science communities because it provides a rigorous foundation for theorem proving. Recent advancements in the field provide foundation models and sophisticated agentic systems pushing the boundaries of formal mathematical reasoning to approach the natural language capability of LLMs (Chen et al., 2025b). However, little attention has been given to the formal physics reasoning, which also heavily relies on similar problem-solving and theorem-proving frameworks. To solve this problem, this paper presents, to the best of our knowledge, the first approach to enhance formal theorem proving in the physics domain. We compose a dedicated dataset **PhysLean-Data** for the task. It is composed of theorems sampled from PhysLean (Tooby-Smith, 2025) and data generated by a conjecture-based formal data generation pipeline. In the training pipeline, we leverage DeepSeek-Prover-V2-7B, a strong open-source mathematical theorem prover, and apply Reinforcement Learning with Verifiable Rewards (RLVR) to train our model **PhysProver**. Comprehensive experiments demonstrate that, using only ∼5.5K training samples, **PhysProver** achieves an overall **2.4%** improvement in multiple sub-domains. Furthermore, after formal physics training, we observe **1.3%** gains on the MiniF2F-Test benchmark, which indicates non-trivial generalization beyond physics domains and enhancement for formal math capability as well. The results highlight the effectiveness and efficiency of our approach, which provides a paradigm for extending formal provers outside mathematical domains. To foster further research, we will release both our dataset and model to the community.

1 Introduction

Formal reasoning has long been recognized as a cornerstone of human intelligence and a critical domain in machine learning research (Newell and Simon, 1956). With the recent advancements in Large Language Models (LLMs), much research has investigated their applications in formal theorem proving. They explored domains from training foundation models (Lin et al., 2025b; Ren et al., 2025; Wang et al., 2025c) and specialized agent framework (Wang et al., 2025d; Chen et al., 2025b; Varambally et al., 2025). Among these, math theorem proving in Lean4 (Moura and Ullrich, 2021a) has emerged as one of the most extensively studied areas (Wang et al., 2024; Lin et al., 2025a; Xin et al., 2024). Researchers typically start from a general-purpose LLM, employing Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) to enhance the formal reasoning capability. This approach has achieved strong results on formal math benchmarks, such as MiniF2F (Zheng et al., 2022) and PutnamBench (Tsoukalas et al., 2024).

Previous works have demonstrated that developing expert models for Lean4 theorem proving demands substantial training data and a large amount of GPU hours. For instance, DeepSeek-Prover (Xin et al., 2024) applies a 120B math-related tokens continue pretraining and 8M formal statements with proofs to train an expert prover. Similarly, Goedel-Prover (Lin et al., 2025a) applies expert iteration on more than 1 million formal statements. Despite these advancements, formal theorem proving faces significant challenges due to a scarcity of high-quality data that is able to give the model a general formal reasoning capability, rather than focusing on a narrow field (Li et al., 2025).

While significant progress has been made in mathematical theorem proving, the formal physics domain remains largely overlooked. Physics, with its reliance on rigorous mathematical foundations and formal derivations, offers a natural yet under-explored extension to formal reasoning. Li et al. (2025) highlights that SOTA theorem-proving models perform poorly in physics-related tasks but fail



(a) Data Generation Stage (b) Self-Evolving Stage

Figure 1: **Physics Prover Framework**: (a) Data Generation Stage: the training set comprises 5,541 physics statements from both PhysLean (Tooby-Smith, 2025) and synthetic lemmas from Claude-4.5-Sonnet, where the latter are further filtered by Lean syntax and proof existence checks. (b) Self-Evolving Stage: after obtaining the training set, GRPO (Shao et al., 2024) is adopted to train the base prover models, with reward signals of proof correctness provided by Lean.

to propose methods for improvement.

To settle this gap, we, as far as we are concerned, take the first step toward enhancing theorem proving in the physics domain by constructing a specialized data pipeline and employing Reinforcement Learning with Verifiable Rewards (RLVR). The overview of our framework can be found in Figure 1. Specifically, we collect foundational theorems and lemmas from the open-source repository of PhysLean (Tooby-Smith, 2025), which contains Lean4-based results across advanced physics domains such as Quantum Field Theory and String Theory. The extracted data, along with their headers, are divided into the training and testing sets. To augment the training dataset, we apply Claude-4.5 to generate additional conjectures based on the dataset. Subsequently, we apply formal LLMs to annotate these conjectures, thereby formulating the Basic Physics Lean training dataset, which contains approximately 5.5K training samples and 250 testing samples.

With the dataset, we leverage RLVR (Lambert et al., 2025) to enhance physical theorem-proving capability using the GRPO algorithm. Our evaluation demonstrates consistent improvement across multiple physics domains and achieves a **2.4%** of overall improvements compared to SOTA math provers on the testing dataset. Furthermore, when tested on the Out-of-Distribution (OOD) MiniF2F benchmark (Zheng et al., 2022), **PhysProver** achieved over 1% of improvement compared to the base model under pass@16. It demonstrates the effectiveness of our approach and shows that physics dataset training can enhance the formal math capability of the model.

We summarize our contribution as follows:

1. **Physics-specific conjecture synthesis pipeline.** We design and release the first data generation pipeline tailored to formal physics theorem proving, addressing non-trivial challenges unique to physics: header-preserving conjecture generation, the failure of off-the-shelf auto-formalizers due to diverse physical header dependencies, and multi-prover verification to ensure coverage across proof styles.

2. **PhysLeanData and PhysProver.** We construct **PhysLeanData**, a compact high-quality dataset of 5,541 physics theorems in Lean4, and train **PhysProver** via RLVR, achieving consistent $+2.4\%$ improvements over SOTA formal math provers on our physics test set.

3. **Surprising math-transfer finding.** Despite training exclusively on physics theorems, **PhysProver** improves over its base model on MiniF2F-Test by $+1.3\%$—counter to the

expected alignment tax—revealing an unexpected positive transfer from physics to mathematics theorem proving.

2 Related Works

2.1 Formal Math Reasoning

Formal math reasoning involves representing mathematical components in a computer-verifiable format. It reduces the ambiguity and establishes a rigorous foundation for logical reasoning. Over the past decades, researchers have developed numerous Formal Languages (FLs) based on two primary theoretical frameworks. The first relies on dependent type languages, such as Lean (De Moura et al., 2015; Moura and Ullrich, 2021b) and Coq (Coq, 1996), where formal verification is achieved through a small kernel to perform type checking. The second line utilizes higher-order logic to quantify functions and predicates. This line of work is represented by languages such as Isabelle (Paulson, 1994), HOL, and HOL Light (Harrison, 2009). Among the above languages, Lean4 (Moura and Ullrich, 2021b) has gained significant attention due to its expressiveness and extensive Mathlib4 repository, which encompasses almost all major mathematical domains.

The rise of LLMs has accelerated the advancements in formal proving tasks. Researchers have compiled extensive datasets of mathematical theorems and proofs(Wang et al., 2025c; Lin et al., 2025a; Dong and Ma, 2025), which provide a robust foundation for model training. Building on these resources, increasingly sophisticated models have emerged. Early efforts, such as Expert Iteration (Polu et al., 2022), employed iterative annotation using LLMs to enhance the training data. Open-source frameworks like DeepSeek-Prover (Xin et al., 2024) and Theorem-Llama (Wang et al., 2024) further advanced the formal provers. More recently, RLVR has enabled Long CoT training for formal theorem proving, which works like MA-LoT (Wang et al., 2025c), Kimina-Prover (Wang et al., 2025a), DeepSeek-Prover-V2 (Ren et al., 2025), and Goedel-Prover-V2 (Lin et al., 2025b), achieving notable progress. The emergence of agentic frameworks, such as Hilbert (Varambally et al., 2025) and Seed-Prover-V1 (Chen et al., 2025c), achieves notable progress by enabling multi-agent theorem decomposition and sub-goal proofs. The latest works apply agentic RL to push LLMs' formal reasoning capability closer to natural language proficiency (Chen et al., 2025b). Despite these advancements, formal reasoning in physics remains an underexplored domain, representing a significant opportunity for future research.

2.2 LLM for Physics Reasoning

With the rapid development of general reasoning capabilities in LLMs, researchers are actively exploring the application of these models in more diverse fields (Wang et al., 2025b). Among them, physics reasoning is one key field that receives significant attention. In the context of benchmarks, early comprehensive benchmarks, such as SciBench (Wang et al., 2023) and GPQA (Rein et al., 2024), evaluate college-level scientific problem-solving across multiple scientific fields, including physics. More recently, physics benchmarks have emerged at multiple difficulty levels: UGPhysics (Xu et al., 2025) presents 5,520 undergraduate-level bilingual problems that advanced thinking models are hard to solve; OlympiadBench (He et al., 2024) introduces 8,476 Olympiad-level problems with multi-module inputs; and recent HiPhO (Yu et al., 2025) compiles the latest 13 Physics Olympiad exams in 2024-2025 with human-aligned evaluation.

On the model training side, researchers began exploring the potential for LLMs as physics reasoning tools from an early stage. Early works have demonstrated that LLMs can solve complex word problems that require calculation and inference (Ding et al., 2023). Such capability can be further enhanced by Reinforcement Learning from Human Feedback (RLHF) (Anand et al., 2024) or simple multi-agent collaboration (Pang et al., 2025). Recent works apply RLVR on natural language physics problems, with P1 (Chen et al., 2025a) achieving gold-level IPhO performance. However, with a lack of datasets and training methods, developing LLMs for formal physics reasoning is relatively understudied currently (Li et al., 2025).

3 Methodology

3.1 Seed Dataset Collection

We construct a lemma–proof dataset from the PhysLean GitHub repository (Tooby-Smith, 2025) by extracting all provable lemmas from `.lean` files along with their preceding formal headers. The lemma statements with context serve as inputs, while the corresponding proof scripts serve as outputs. We filter the samples to retain only those with

a total length under 4,096 tokens. The resulting corpus contains over 3,000 examples, which are randomly split into training and test sets at approximately a 9:1 ratio, yielding 2,933 training and 250 test instances. The dataset spans a broad range of domains in physics and mathematics, encompassing classical and modern physics (e.g., classical mechanics, electromagnetism, quantum mechanics, and relativity) as well as advanced theoretical areas such as quantum field theory, string theory, and mathematical foundations. An example of the collected data is shown in Figure 6. A discussion of the key differences between physics and mathematics theorem proving in Lean4 is provided in Appendix C.

3.2 Synthetic Data Generation

To augment our dataset, we construct a conjecture generation and verification pipeline inspired by STP (Dong and Ma, 2025). Specifically, we treat our initial data as seed data, denoted as $D_{\text{seed}} = \{(h_i, l_i, p_i)\}_{i=1}^{N}$, where h_i, l_i, and p_i are the header, lemma, and corresponding proof of the ith sample, and N is the total number of seed examples. For each sample, we use Claude-4.5-Sonnet (Anthropic, 2025) to generate 10 conjectures by providing the header–lemma pairs (h_i, l_i), yielding 29,330 candidate statements. The prompting template is provided in Figure 10 in the Appendix.

After collecting the conjectures, we apply a two-stage pipeline to select well-formed and correct statements. We first examine the **syntactic correctness** of each conjecture. Specifically, for each conjecture c_{ij}, we append it to the corresponding header h_i to form $D_c = \{(h_i, c_{ij}) \mid i = 1, \ldots, N,\ j = 1, \ldots, 10\}$ and use the Lean verifier to check whether the statement is well-formed. This includes verifying that all variables are properly defined and that all referenced definitions and theorems exist. After this step, 6,971 conjectures remain, corresponding to a retention rate of 23.8%.

The second stage examines the **provability** of conjectures. Given a conjecture with its corresponding header, we leverage DeepSeek-Prover-V2-7B (Ren et al., 2025), Kimina-Prover-Distill-8B (Wang et al., 2025a), and Goedel-Prover-V2-8B (Lin et al., 2025b), to generate 16 proofs, producing response samples $\{(h_i, c_{ij}, r_p)\}_{p=1}^{16}$. A conjecture is deemed provable if

$$\exists\, p,\ 1 \le p \le 16 : \texttt{Verify}(h_i, c_{ij}, r_p) = \texttt{True}$$

where `Verify` denotes the Lean verification result. This process yields 2,608 verified conjectures, representing an overall pipeline yield rate of 8.9%, which is comparable to STP (Dong and Ma, 2025). Combining these with the 2,933 seed training examples in Section 3.1 produces a total of 5,541 training instances for our experiments.

Notably, we also compared different proprietary models, including GPT-5 (OpenAI, 2025) and Gemini-2.5-Pro (Google, 2025). However, the syntactically correct rates of their generated conjectures were substantially lower than those produced by Claude. We additionally explored the approach of generating conjectures in natural language and converting them to Lean4 statements using an auto-formalizer. However, auto-formalizers fail at this task due to the difficulty of identifying a uniform header, which is caused by the complex dependencies in physical statements. Consequently, this approach also yielded low success rates. We further compared against the open-source Qwen-QwQ-32B (Team, 2025), which achieved only a 0.5% conjecture yield rate vs. 8.9% for Claude-4.5-Sonnet; the full comparison is in Appendix J.

3.3 Self-Evolving Pipeline

We conduct Reinforcement Learning (RL) to lift the performance on physics domain. Specifically, our experiments are mainly based on Group Relative Policy Optimization (GRPO) (Shao et al., 2024). The full GRPO objective and derivation are given in Appendix B. In our setting, for each prompt x in the training set, G responses are sampled during rollout and optimized with group-normalized advantages and a clipped importance ratio.

The reward signal $r(x, y_i)$ is provided by the Lean verifier to guide the reinforcement learning process. Specifically, a score of 1 or 0 is presented to indicate whether the proof is correct or not. Because of the symbolic nature of Lean, all verified proofs with reward 1 are correct, with no hallucination at all, which allows the model to learn the foundation of physics in a concrete and rigorous way. To further reduce the difficulty of the learning process, curriculum learning is employed, where the input statements (conjectures) are sorted based on their ground-truth proof lengths. This enables easy-to-hard learning, encouraging prover models to proceed in a bottom-up manner.

Method	Classical	Particle & String	Relativity	Quantum Field Theory	Overall
Proprietary Models					
GPT-5 (OpenAI, 2025)	37.3%	13.4%	21.3%	35.2%	26.4%
Claude-4.5-Sonnet (Anthropic, 2025)	52.9%	19.4%	29.5%	**39.4%**	34.4%
Formal Math Provers					
Kimina-Prover-Distill-8B (Wang et al., 2025a)	35.3%	14.9%	29.5%	22.5%	24.8%
Goedel-Prover-V2-8B (Lin et al., 2025b)	49.0%	19.4%	34.4%	28.2%	31.6%
Deepseek-Prover-V2-7B (Ren et al., 2025)	54.9%	23.9%	37.7%	25.4%	34.0%
Formal Physics Provers					
PhysProver	**58.8% (+3.9%)**	**26.9% (+3.0%)**	**39.3% (+1.6%)**	26.8% (+1.4%)	**36.4% (+2.4%)**
Ablation					
PhysProver (Random Order)	56.9%	23.9%	37.7%	29.6%	35.6%

Table 1: **Main experimental results**. We evaluate all the models on **PhysLeanData** test set, which includes Classical, Particle & String, Relativity, and Quantum Field Theory domains. The pass@16 accuracy is reported. The "Random Order" row ablates curriculum learning by replacing easy-to-hard proof-length ordering with random ordering of the same training set.

4 Experiments

For a more controlled comparison between DeepSeek-Prover-V2-7B and **PhysProver** with statistical reliability, we report mean $\pm$ std results over 3 independent runs in Appendix H.

To evaluate our methodology, we use **PhysLeanData** to train popular Lean-based formal mathematics provers. Our experiments reveal that strong mathematical reasoning models exhibit notable limitations when handling formal physics problems, underscoring the importance of domain-specific formal datasets and self-evolving strategies.

4.1 Experimental Setup

4.1.1 Dataset and Tasks

Model performance is evaluated on the test set of **PhysLeanData**, which shares the same source as the training set with a 9:1 train-test split. To ensure fair comparison across models with different context lengths, we retain only samples with prompt lengths under 4,096 tokens, resulting in 250 lemmas in the final evaluation set.

For finer-grained analysis, we organize the test samples into four physics categories: Classical & Foundational Physics, Particle & String Physics, Relativity & Spacetime, and Quantum Field Theory. This classification reflects distinct theoretical frameworks and varying levels of required domain expertise. Further details are provided in Appendix D.

4.1.2 Models and Baselines

We compare several popular open-source prover models, including DeepSeek-Prover-V2-7B (Ren et al., 2025), Kimina-Prover-Distill-8B (Wang et al., 2025a), and Goedel-Prover-V2-8B (Lin et al., 2025b), all of which are strong formal theorem provers tailored for mathematical domains. Since DeepSeek-Prover-V2-7B performs the best among them, our experiments will focus on training the DeepSeek prover to push the boundaries of open-source models.

For baselines, we first report the performance of DeepSeek-Prover-V2-7B, Kimina-Prover-Distill-8B, and Goedel-Prover-V2-8B without any additional training. We also include comparisons with strong proprietary systems, namely GPT-5 (OpenAI, 2025) and Claude-4.5-Sonnet (Anthropic, 2025). For all baselines, we use a fixed sampling budget and report pass@16 accuracy, ensuring fair comparison under a consistent inference budget. For open-source provers, we use the prompt template provided in Appendix L.1. For proprietary models, we employ a tailored Chain-of-Thought (CoT) (Wei et al., 2023) prompt to encourage step-by-step reasoning before generating the final proof.

4.2 Implementation Details

We directly apply Reinforcement Learning starting from the DeepSeek-Prover-V2-7B using verl (Sheng et al., 2025). Specifically, we apply GRPO with rule-based rewards (Lambert et al., 2025; DeepSeek-AI et al., 2025) to guide the self-evolving training. In particular, the Lean verifier with version `4.20.0` is integrated into the verl framework for verifying the proofs. The reward score for each trajectory is calculated as follows:

$$r(x, y_i) = \begin{cases} 1 & \text{if Verify}(x, y_i) = \text{True} \\ 0 & \text{otherwise} \end{cases}$$

Additionally, if the proof contains `sorry`, `admit`, or `apply?` keywords, we directly assign a 0 for the

reward score to avoid reward hacking. Furthermore, to allow a smooth transition of difficulties during the learning process, curriculum learning (Parashar et al., 2025) is employed by sorting the lemma based on their ground-truth proof lengths.

We train all models on 8×H200 GPUs with a constant learning rate of $1e^{-6}$ and a batch size of 256 for 2 epochs, where the training takes approximately 8 hours. Notably, we do not have a warm-up stage with Supervised Fine-Tuning (SFT) because it degrades performance. This behavior is investigated and further analyzed in Section 6. We also investigate the Rejection-Sampling Fine-tuning method (Yuan et al., 2023; Dong et al., 2023) in Section 6.

4.3 Experiment Results

Our experimental results are presented in Table 1. We first observe that all existing models achieve relatively low scores despite their proficiency in mathematical theorem proving, with none exceeding 40% accuracy. Notably, even small open-source theorem prover models exhibit competitive accuracy compared to the latest proprietary systems, such as Claude-4.5-Sonnet and GPT-5. However, proprietary models demonstrate different strengths across physical domains compared to their open-source counterparts. For instance, all open-source provers achieve below 30% accuracy on Quantum Field Theory, whereas proprietary models exceed 35%. This suggests that proprietary and open-source models may be trained on different mixtures of physics data. We also investigated context length in the Quantum Field Theory category and found that the average length is one-third longer than in other domains. These findings align with those of Li et al. (2025), suggesting that larger models such as Claude demonstrate superior in-context learning ability, thereby achieving better performance than open-source models.

Our trained model, PhysProver, substantially surpasses its formal mathematics prover counterparts, consistently achieving gains across all categories. Specifically, on the most challenging domains—Particle & String Physics—where all baselines exhibit low accuracy, our model still yields a notable improvement of 3.0%. A fine-grained breakdown of the sub-domains is provided in Appendix D. These results demonstrate the effectiveness of extending a mathematics prover to physics domains with only small high-quality datasets. Moreover, the continued performance gains sug-

Header and Context Lemmas

```
import PhysLean.QFT.PerturbationTheory.WickAlgebra.
      NormalOrder.Lemmas
import PhysLean.QFT.PerturbationTheory.WickAlgebra.
      TimeOrder

namespace FieldSpecification
variable {𝓕 : FieldSpecification}

def timeContract (φ ψ: 𝓕.FieldOp) : 𝓕.WickAlgebra
      :=
    𝒯(ofFieldOp φ* ofFieldOp ψ) - 𝒯(ofFieldOp φ*
          ofFieldOp ψ)

lemma timeContract_eq_superCommute (φ ψ: 𝓕.FieldOp)
       :
    timeContract φψ= if timeOrderRel φψthen [anPart
          φ, ofFieldOp ψ]ₛ
    else 𝒮(𝓕 |>ₛ φ, 𝓕|>ₛ ψ) ·[anPart ψ, ofFieldOp
          φ]ₛ := by
  split_ifs
  ·rename_i h
    rw [timeContract_of_timeOrderRel _ _ h]
  ·rename_i h
    rw [timeContract_of_not_timeOrderRel_expand _ _ h
          ]

lemma timeContract_zero_of_diff_grade (φ ψ: 𝓕.
      FieldOp) (h : (𝓕 |>ₛ φ) ≠(𝓕 |>ₛ ψ)) :
    timeContract φψ= 0 := by
  by_cases h1 : timeOrderRel φψ
  ·rw [timeContract_of_timeOrderRel _ _ h1]
    rw [
          superCommute_anPart_ofFieldOpF_diff_grade_zero
          ]
    exact h
  ·rw [timeContract_of_not_timeOrderRel _ _ h1]
    rw [timeContract_of_timeOrderRel _ _ _]
    rw [
          superCommute_anPart_ofFieldOpF_diff_grade_zero
          ]
    simp only [instCommGroup.eq_1, smul_zero]
    exact h.symm
    have ht := IsTotal.total (r := 𝓕.timeOrderRel) φ
          ψ
    simp_all
    .......
```

DeepseekProver + PhysProver

```
/-- This lemma claims that the normal ordering of the
       time-contracted product of any two field
      operators φand ψis identically zero.-/
lemma normalOrder_timeContract (φ ψ: 𝓕.FieldOp) :
    𝒩(timeContract φψ) = 0 := by
  rw [timeContract_eq_superCommute]
  split_ifs <;> simp_all [timeContract,
        superCommute_anPart_ofFieldOpF_diff_grade_zero
        ,
    smul_eq_zero, map_zero]
```

DeepseekProver

```
lemma normalOrder_timeContract (φ ψ: 𝓕.FieldOp) :
    𝒩(timeContract φψ) = 0 := by
  rw [timeContract_eq_superCommute]
  split_ifs
  ·rw [normalOrder_ofFieldOp_pair_eq_zero]
    <;> simp_all [timeOrderRel_of_isContraction]
  ·rw [normalOrder_anPart_ofFieldOp_smul]
    <;> simp_all [timeOrderRel_of_isContraction]
```

Figure 2: Successful examples from the **PhysProver** and failed proofs from the base model for the same statements. **PhysProver** demonstrates better in-context learning ability to make good usage of lemmas.

Problem Category		DeepSeek-V2 Pass@16	DeepSeek-V2 + PhysLeanData Pass@16
Olympiad	IMO	20.0%	20.0%
	AIME	53.3%	46.7%
	AMC	55.6%	55.6%
MATH	Algebra	90.0%	92.9%
	Number Theory	85.0%	88.3%
Custom	Algebra	44.4%	44.4%
	Number Theory	50.0%	50.0%
	Induction	50.0%	50.0%
Overall		68.4%	**69.7%**

Table 2: **OOD Generalization** on MiniF2F-Test (Zheng et al., 2022) (pass@16).

gest that current provers are far from saturated, indicating that constructing high-quality physics-specific datasets remains a promising direction

On top of that, the small 7B-sized PhysProver model outperforms both GPT-5 and Claude-4.5-Sonnet in terms of overall performance, which shows the huge potential of small expert models in specific domains of formal physics theorem proving. This provides a promising path toward efficient training of physics prover models. The ablation row in Table 1 shows that curriculum learning (easy-to-hard ordering by proof length) contributes $+0.8\%$ overall vs. random ordering, with meaningful gains in Particle & String ($+3.0\%$) and Relativity ($+1.6\%$) suggesting that progressive difficulty is especially beneficial for structurally complex proofs. An error analysis of failure cases is provided in Appendix G.

5 Analysis

5.1 Improved In-Context Learning Through Reinforcement Learning

In this subsection, we provide a detailed analysis of the performance gains achieved by **PhysProver** through a comparative examination of proofs generated by the baselines and our model. Figure 2 presents an illustrative example from our test set along with the corresponding generations. The header and lemmas constitute the context for physical theorem proving, where the lemmas serve as auxiliary tools during the proof process.

We observe that **PhysProver** consistently makes correct use of functions and lemmas, with successful applications highlighted in blue. For instance, to prove the given conjecture, it first applies *timeContract_eq_superCommute*, followed by the function *timeContract*. Subsequently, the model correctly invokes *superCommute_anPart_ofFieldOpF_diff_grade_zero*, demonstrating effective utilization of contextual information. By synthesizing the knowledge provided in the context, **PhysProver** successfully completes the proof.

In contrast, while the base model initially applies *timeContract_eq_superCommute* correctly, it subsequently generates hallucinated content, including non-existent lemmas such as *normalOrder_ofFieldOp_pair_eq_zero* and *timeOrderRel_of_isContraction* (marked in red). These observations suggest that the reinforcement learning process on **PhysLeanData** enhances performance by enabling the model to better leverage contextual information and comprehend domain-specific terminology. This finding also accounts for the low accuracy observed across all base models: their unfamiliarity with physics-specific lemmas and contextual structures impedes their ability to effectively utilize these resources for proof completion. Additional case studies and trajectory-level pass rate analysis are provided in Appendix I.

5.2 Out-of-Distribution Generalization

Surprisingly, despite the typical *alignment tax* (Lin et al., 2024) from domain-specific fine-tuning, training on physics problems improves out-of-distribution performance on MiniF2F-Test (Zheng et al., 2022)—a 244-statement Lean4 math benchmark spanning high school competition to undergraduate-level proofs. We follow the evaluation setup of Ren et al. (2025) and report pass@16 using the standard DeepSeek prompt template.[1]

As shown in Table 2, **PhysProver** surpasses the baseline by $+1.3\%$ overall. Gains concentrate in MATH Algebra ($+2.9\%$) and Number Theory ($+3.3\%$), indicating that physics training transfers foundational algebraic reasoning. However, harder Olympiad-level problems (e.g., AIME) do not benefit, suggesting those require capabilities beyond what physics-based training provides. A detailed error breakdown is in Appendix K.

6 Revisiting the Role of Supervised Fine-tuning

We additionally investigated whether conducting Supervised Fine-tuning (SFT) on **PhysLeanData** could enhance model performance on Physics, following standard practice in training specialized LLMs. However, we did not observe any improve-

[1] https://huggingface.co/deepseek-ai/DeepSeek-Prover-V2-7B

Method	Budget	Classical	Particle & String	Relativity	Quantum Field Theory	Overall
Deepseek-Prover-V2-7B	pass@16	**54.9%**	23.9%	37.7%	25.4%	34.0%
Deepseek-Prover-V2-7B + Phys SFT	pass@16	45.1% (-9.8%)	19.4% (-4.5%)	26.2% (-11.5%)	23.9% (-1.5%)	27.6% (-6.4%)
Deepseek-Prover-V2-7B + Phys RAFT	pass@16	52.9% (-2.0%)	**25.4%** **(+1.5%)**	**41.0%** **(+3.3%)**	**28.2%** **(+2.8%)**	**35.6%** **(+1.6%)**

Table 3: **Supervised Fine-tuning (SFT) and Rejection-Sampling Fine-tuning (RAFT) on PhysLeanData** of Deepseek-Prover-V2-7B. The pass@16 accuracy drops significantly after SFT, but increases after RAFT.

	Training Set	Test Set
DS-Prover-SFT	1.817	1.711
DS-Prover-RAFT	1.321	**1.186**
DS-Prover-GRPO	**1.141**	1.209

Table 4: Perplexity on the training set and the test set for DS-Prover-SFT, DS-Prover-RAFT, and DS-Prover-GRPO.

ment on our test set after SFT. Instead, we observed consistent performance degradation.

Specifically, we first fine-tuned Deepseek-Prover-V2-7B on **PhysLeanData**, where ground-truth answers were either extracted from the PhysLean library **written by humans**, or generated by open-source provers with subsequent verifications. The training sample template follows the RL prompt template in L.1, with loss computation restricted to the completion portion. We then consider Rejection Sampling Fine-tuning, or Reward-Ranked Fine-tuning (RAFT) (Dong et al., 2023; Yuan et al., 2023), where we sample Deepseek-Prover-V2-7B on the training set and retain only the correct proofs as our new training set. We fine-tuned Deepseek-Prover-V2-7B on both training sets for one epoch, using a learning rate of $5e^{-7}$ and a batch size of 32. They are denoted as DS-Prover-SFT and DS-Prover-RAFT, respectively.

As shown in Table 3, for the DS-Prover-SFT, we observe consistent performance degradation across all categories, with an average accuracy decline of 6.4%. On the contrary, DS-Prover-RAFT demonstrates an overall 1.6% improvement, with all other 3 categories increasing except Classical Physics. We attribute this performance difference to the distributional properties of the training data. The original **PhysLeanData** consists of human-written examples, which are Out-of-Distribution (OOD) with respect to the model's generation capabilities. In contrast, Rejection Sampling yields In-Distribution (ID) data that more closely aligns with the model's output distribution. Consequently, ID data may be easier for the model to learn from, leading to improved performance.

To understand this performance difference, we conduct perplexity-based probing experiments. For each model, we sample $K{=}16$ responses per prompt and compute the mean perplexity over 50 randomly selected prompts from each of the training and test sets; lower perplexity indicates the model generates responses it considers likely, suggesting lower self-uncertainty. Formal definitions of $\overline{PPL}(x)$ and $PPL(y)$ are provided in Appendix F.

As shown in Table 4, DS-Prover-GRPO and DS-Prover-RAFT yield substantially lower perplexity than DS-Prover-SFT on both training and test sets. This indicates that SFT paradoxically raises model uncertainty for a model already pre-trained on domain-specific data, while RL and rejection-sampling better align training with the model's own output distribution, reducing uncertainty and improving accuracy. We also explore Reinforcement Learning after Rejection-Sampling Fine-tuning in Appendix E, but do not observe improvements.

7 Conclusion

In this paper, we present the first systematic effort to advance formal theorem proving in the physical domain. We first introduce **PhysLeanData**, a dataset of physical theorems formalized in Lean4, along with a conjecture formulation pipeline for generating valid and correct conjectures. By applying Reinforcement Learning with Verifiable Rewards (RLVR) to an open-source state-of-the-art theorem prover, our **PhysProver** achieves consistent **2.4%** improvements across physical subdomains such as Quantum Field Theory using only 5.5K samples. The model also demonstrates over **1%** improvement on the out-of-distribution MiniF2F-test benchmark, highlighting strong generalization capability. Our work bridges a critical gap between formal theorem proving in mathematics and its application to the physical sciences. We will publicly release our dataset and models to facilitate future research in this direction.

8 Limitations

Our work has three main limitations: (1) a compute-intensive conjecture pipeline with an 8.9% yield rate that restricts scale, and a dataset sourced solely from PhysLean that may underrepresent some subdomains; (2) the absence of a competition-level formal physics benchmark, making precise OOD evaluation difficult; and (3) infeasibility of autoformalization from natural language to Lean4, due to the diversity of physics-specific header definitions required. Extended discussion is in Appendix A.

References

Avinash Anand, Kritarth Prasad, Chhavi Kirtani, Ashwin R Nair, Mohit Gupta, Saloni Garg, Anurag Gautam, Snehal Buldeo, and Rajiv Ratn Shah. 2024. Enhancing llms for physics problem-solving using reinforcement learning with human-ai feedback. *arXiv preprint arXiv:2412.06827*.

Anthropic. 2025. Claude sonnet 4.5 system card. Technical report, Anthropic.

Jiacheng Chen, Qianjia Cheng, Fangchen Yu, Haiyuan Wan, Yuchen Zhang, Shenghe Zheng, Junchi Yao, Qingyang Zhang, Haonan He, Yun Luo, and 1 others. 2025a. P1: Mastering physics olympiads with reinforcement learning. *arXiv preprint arXiv:2511.13612*.

Jiangjie Chen, Wenxiang Chen, Jiacheng Du, Jinyi Hu, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Wenlei Shi, and 1 others. 2025b. Seed-prover 1.5: Mastering undergraduate-level theorem proving via learning from experience. *arXiv preprint arXiv:2512.17260*.

Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, and 1 others. 2025c. Seed-prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*.

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

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 2015. 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*, pages 378–388. Springer.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jingzhe Ding, Yan Cen, and Xinyuan Wei. 2023. Using large language model to solve and explain physics word problems approaching human level. *arXiv preprint arXiv:2309.08182*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *Preprint*, arXiv:2304.06767.

Kefan Dong and Tengyu Ma. 2025. Stp: Self-play llm theorem provers with iterative conjecturing and proving. *Preprint*, arXiv:2502.00212.

Google. 2025. Gemini 2.5: Our newest gemini model with thinking. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/. Accessed: 2025.

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

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

Yuxin Li, Minghao Liu, Ruida Wang, Wenzhao Ji, Zhitao He, Rui Pan, Junming Huang, Tong Zhang, and Yi R Fung. 2025. Lean4physics: Comprehensive reasoning framework for college-level physics in lean4. *arXiv preprint arXiv:2510.26094*.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, Hanze Dong, Renjie Pi, Han Zhao, Nan Jiang, Heng Ji, Yuan Yao, and Tong Zhang. 2024. Mitigating the alignment tax of rlhf. *Preprint*, arXiv:2309.06256.

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou

Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. 2025a. Goedel-prover: A frontier model for open-source automated theorem proving. *Preprint*, arXiv:2502.07640.

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, and 1 others. 2025b. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*.

Leonardo de Moura and Sebastian Ullrich. 2021a. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635, Cham. Springer International Publishing.

Leonardo de Moura and Sebastian Ullrich. 2021b. 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*, pages 625–635. Springer.

Allen Newell and Herbert Alexander Simon. 1956. *The Logic Theory Machine: A Complex Information Processing System*. RAND Corporation, Santa Monica, CA.

OpenAI. 2025. GPT-5 system card. Technical report, OpenAI.

Xinyu Pang, Ruixin Hong, Zhanke Zhou, Fangrui Lv, Xinwei Yang, Zhilong Liang, Bo Han, and Changshui Zhang. 2025. Physics reasoner: Knowledge-augmented reasoning for solving physics problems with large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11274–11289.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. 2025. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *Preprint*, arXiv:2506.06632.

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

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

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, and 1 others. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297. ACM.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Joseph Tooby-Smith. 2025. Heplean: Digitalising high energy physics. *Computer Physics Communications*, 308:109457.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. *Preprint*, arXiv:2407.11214.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. 2025. Hilbert: Recursively building formal proofs with informal reasoning. *Preprint*, arXiv:2509.22819.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, and 1 others. 2025a. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*.

Ruida Wang, Yuxin Li, Tong Zhang, and 1 others. 2025b. Let's reason formally: Natural-formal hybrid reasoning enhances llm's math capability. *arXiv preprint arXiv:2505.23703*.

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

Ruida Wang, Jiarui Yao, Rui Pan, Shizhe Diao, and Tong Zhang. 2025d. Gar: Generative adversarial reinforcement learning for formal theorem proving. *arXiv preprint arXiv:2510.11769*.

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

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2023. Scibench: Evaluating college-level scientific

problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *Preprint*, arXiv:2405.14333.

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

Fangchen Yu, Haiyuan Wan, Qianjia Cheng, Yuchen Zhang, Jiacheng Chen, Fujun Han, Yulun Wu, Junchi Yao, Ruilizhen Hu, Ning Ding, and 1 others. 2025. Hipho: How far are (m) llms from humans in the latest high school physics olympiad benchmark? *arXiv preprint arXiv:2509.07894*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *Preprint*, arXiv:2308.01825.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *Preprint*, arXiv:2109.00110.

A Extended Limitations Discussion

Scale and Dataset Coverage. Computational constraints limited the scale of our conjecture generation pipeline: with a yield rate of only 8.9%, most generated conjectures are filtered out during validity and provability verification, and further scaling would require substantially more compute. Our dataset is also derived solely from the PhysLean repository, which may underrepresent certain physics sub-domains.

Benchmarks. No competition-level formal physics benchmark currently exists. While MiniF2F provides rigorous OOD evaluation for mathematics, an analogous physics benchmark grounded in advanced physical theory would enable more precise measurement of formal physics reasoning capability.

Auto-Formalization. We explored but ultimately abandoned auto-formalization from natural language physics statements to Lean4. Physics proofs require highly diverse header definitions (unit systems, coordinate conventions, field operators) that make it infeasible to identify a uniform header for an auto-formalizer—unlike pure mathematics, where Mathlib provides a standardized namespace.

B GRPO Formulation

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a policy gradient method for LLM training. For each prompt x, it samples G responses $\{y_i\}_{i=1}^{G}$ from the old policy and optimizes:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \\ \Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(w_{i,t}(\theta)\,\hat{A}_i, \\ \text{clip}\big(w_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\big)\hat{A}_i\Big)\Bigg] \\ -\beta\,\mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\theta_{\text{ref}}})\,,\end{aligned}$$

where $w_{i,t}(\theta) = \pi_\theta(y_{i,t}|x, y_{i,<t})/\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})$ is the importance ratio and

$$\hat{A}_i = \frac{r(x, y_i) - \text{mean}(\{r(x, y_j)\}_{j=1}^{G})}{\text{std}(\{r(x, y_j)\}_{j=1}^{G})}$$

is the group-normalized advantage. All tokens in y_i share the same advantage $\hat{A}_i$.

C Why Physics Theorem Proving Differs from Mathematics

PhysLean theorems are structural and theoretical rather than numerical: even a statement such as `minkowskiMatrix * minkowskiMatrix = 1` is proved purely symbolically using group-theoretic and algebraic identities, with no floating-point computation. Three factors distinguish physics proving from standard Mathlib mathematics proving:

1. **Domain-specific definitions.** Constructs such as field operators, Lagrangians, and gauge groups are defined locally within PhysLean and do not appear in Mathlib. The prover must reason from first principles within the codebase, without access to a general-purpose lemma library for these objects.

2. **Contextual headers.** Each physics theorem is preceded by a header that introduces custom

namespace definitions absent from general mathematical libraries. This header changes substantially across lemmas, forming a unique proof context that the model must interpret anew for each statement.

3. **Chained physics-specific lemmas.** Proofs routinely chain multiple physics-specific auxiliary lemmas (e.g., properties of the Lorentz group or Wick algebra) within a single unified theoretical framework, requiring the model to retrieve and compose domain-specific facts that are not available through Mathlib lookup.

These distinctions explain why general-purpose mathematical theorem provers achieve strong results on MiniF2F yet struggle on PhysLean problems, and motivate the need for a physics-specific training dataset and pipeline.

D PhysProver Categories

The Classical & Foundational Physics category groups core undergraduate-level subjects, including mathematical methods, classical mechanics, quantum mechanics, statistical mechanics, and electromagnetism. These areas represent foundational discoveries in physics and are primarily textbook-driven, with standardized problem formulations and solution methods.

Particle and String Physics is grouped separately to capture topics centered on high-energy physics and fundamental interactions, often motivated by experimental programs such as those at the Large Hadron Collider. String theory topics are included in this category due to their close conceptual alignment with high-energy theoretical frameworks.

Quantum Field Theory and Relativity are treated as distinct categories due to their advanced mathematical structure and conceptual complexity. Both subjects are typically introduced at the graduate level, with quantum field theory extending quantum mechanics and relativity providing a foundational framework for spacetime and gravitation.

E Reinforcement Learning after Rejection-Sampling Fine-tuning

In this section, we investigate the effectiveness of Reinforcement Learning (RL) following Rejection-Sampling Fine-tuning (RAFT). Starting from the RAFT checkpoint described in Section 6, we apply RL using the same hyperparameters and settings as in our main experiments (Section 4.2). We save checkpoints every 5 steps and report RL performance in Table 5.

We observe that overall performance does not improve beyond 30 steps of GRPO training, with overall accuracy remaining unchanged (both 35.6%). While the Particle & String category shows improvement relative to both the base model and the RAFT model, we observe a corresponding performance drop in the Relativity category, indicating inconsistent gains across categories. Throughout the RL process, both overall and category-level accuracies fluctuate without demonstrating a clear trend. Although we observe the best overall results at Step 20, we attribute this to random noise given the training process. These results suggest that RL after RAFT may not be consistently effective when both stages use the same prompt set. This observation aligns with common practice in LLM training, where different prompt sets are typically employed for fine-tuning and reinforcement learning.

F Perplexity Analysis

To gain a closer look into why SFT raises model uncertainty while GRPO and RAFT reduce it, we compare the uncertainty of DS-Prover-SFT, DS-Prover-RAFT, and DS-Prover-GRPO using average perplexity of sampled responses conditioned on input prompts. Given a prompt x, we sample $K = 16$ responses y_k from the model and compute mean perplexity over 50 randomly selected prompts from each of the training and test sets:

$$\overline{PPL}(x) = \frac{1}{K}\sum_{k=1}^{K} PPL(y^{(k)}), \quad y^{(k)} \sim p_\theta(\cdot \mid x)$$

where

$$PPL(y) = \exp\left(-\frac{1}{|y|}\sum_{t=1}^{|y|} \log p_\theta(y_t \mid y_{<t}, x)\right).$$

Lower perplexity indicates the model generates responses it considers likely; higher values suggest unfamiliarity with the prompt. As shown in Table 4 in the main paper, DS-Prover-GRPO and DS-Prover-RAFT are substantially lower than DS-Prover-SFT on both sets, which explains why GRPO and RAFT improve accuracy while SFT does not. Although supervised fine-tuning directly maximizes target-token likelihood, it does not reduce model uncertainty for a model such as DeepSeek-Prover that has already undergone extensive domain-specific training—suggesting that

Method	Budget	Classical	Particle & String	Relativity	Quantum Field Theory	Overall
DeepSeek-Prover-V2-7B	pass@16	54.9%	23.9%	37.7%	25.4%	34.0%
+RAFT	pass@16	52.9% (-2.0%)	25.4% (+1.5%)	**41.0% (+3.3%)**	**28.2 (+2.8%)**	35.6% (+1.6%)
RAFT+GRPO Step 5	pass@16	52.9% (-2.0%)	23.9% (+0.0%)	**41.0% (+3.3%)**	**28.2 (+2.8%)**	35.2% (+1.2%)
RAFT+GRPO Step 10	pass@16	52.9% (-2.0%)	25.4% (+1.5%)	39.3% (+1.6%)	26.8% (+1.4%)	34.8% (+0.8%)
RAFT+GRPO Step 15	pass@16	**56.9%** (+2.0%)	25.4% (+1.5%)	39.3% (+1.6%)	26.8% (+1.4%)	35.6% (+1.6%)
RAFT+GRPO Step 20	pass@16	54.9% (+0.0%)	25.4% (+1.5%)	**41.0% (+3.3%)**	**28.2 (+2.8%)**	**36.0% (+2.0%)**
RAFT+GRPO Step 25	pass@16	54.9% (+0.0%)	25.4% (+1.5%)	39.3% (+1.6%)	**28.2 (+2.8%)**	35.6% (+1.6%)
RAFT+GRPO Step 30	pass@16	54.9% (+0.0%)	**26.9% (+3.0%)**	37.7% (+0.0%)	**28.2 (+2.8%)**	35.6% (+1.6%)

Table 5: **Reinforcement Learning after Rejection-Sampling Fine-tuning (RAFT) on PhysLeanData** of Deepseek-Prover-V2-7B. The pass@16 accuracy does not improve after RAFT.

RLVR or rejection-sampling methods are preferable for expert-model adaptation in low-resource settings.

G Error Analysis

To understand the sources of failure, we randomly sampled 100 failure cases from **PhysProver**'s test set and manually categorized them. The breakdown is shown in Table 6.

Failure Type	%	Description
Missing Domain Knowledge	48%	Unfamiliarity with physics foundations (e.g., Lorentz group determinant $= \pm 1$)
Tactic Misuse	45%	Wrong tactic applied in context (`aesop`, `simp`, `omega`, `fin_cases`, `rfl`)
Hallucinated Lemmas	7%	Non-existent physics-specific identifiers invoked

Table 6: Error analysis on 100 randomly sampled failure cases from the **PhysProver** test set.

The dominant failure category (48%) reflects the model's unfamiliarity with physics-specific mathematical foundations absent from standard Mathlib-based training—for example, proofs involving properties of the Lorentz group or gauge field operators. The second-largest category (45%) is tactic misuse: the model applies a valid Lean4 tactic in the wrong context (e.g., `omega` on a non-linear arithmetic goal). Only 7% of failures involve hallucinated lemma names, indicating that RLVR on **PhysLeanData** has substantially improved lexical faithfulness to the physics codebase, but deeper semantic understanding of physics domain knowledge remains the primary bottleneck.

H Statistical Significance: Multi-Run Results

Table 7 reports mean $\pm$ std over 3 independent inference runs (DeepSeek-Prover-V2-7B) and 3 independent RL re-runs with different random seeds (PhysProver).

Method	Classical	P&S	Relativity	QFT	Overall
DeepSeek-Prover-V2-7B	$56.2_{\pm 1.1}$	$21.4_{\pm 2.3}$	$36.6_{\pm 0.9}$	$26.3_{\pm 0.8}$	$33.6_{\pm 0.4}$
PhysProver	$\mathbf{57.5}_{\pm 1.1}$	$\mathbf{26.4}_{\pm 0.9}$	$\mathbf{39.3}_{\pm 0.0}$	$27.7_{\pm 0.8}$	$\mathbf{36.3}_{\pm 0.2}$

Table 7: Multi-run results (mean $\pm$ std, pass@16). DeepSeek-Prover-V2-7B: 3 independent inference runs. PhysProver: 3 independent RL training runs with different random seeds. The consistently lower std for PhysProver demonstrates stable training.

I Additional Case Studies and Trajectory-Level Analysis

I.1 Trajectory-Level Pass Rate

To complement the problem-level pass@N metric—which can be sensitive to variance on small per-category test sets—we report trajectory-level accuracy: the fraction of all generated proof attempts (trajectories) that are verified correct. This metric aggregates over all 16 attempts per problem, making it more robust to individual problem variance.

Sub-domain	#Problems	#Trajectories	Baseline	PhysProver
Classical	51	816	42.8%	45.8% (+3.1%)
QFT	71	1,136	15.8%	16.9% (+1.1%)

Table 8: Trajectory-level pass rate. Each problem generates 16 proof trajectories; we report the fraction verified correct across all trajectories. Δ is relative to DeepSeek-Prover-V2-7B.

At the trajectory level, **PhysProver** yields consistent improvements in Classical ($+3.1\%$, $+25$ correct trajectories out of 816) and Quantum Field Theory ($+1.1\%$, $+13$ correct trajectories out of 1,136), demonstrating that the gains are systematic rather than attributable to stochastic variation on a handful of problems.

I.2 Additional Case Studies

Beyond the `normalOrder_timeContract` example in Figure 2, we present three further cases in Figures 3–5 where **PhysProver** succeeds and the

baseline fails, illustrating proof-level semantic improvements: the model learns correct names and application patterns for physics-specific and Lean-standard lemmas rather than memorizing superficial token patterns.

J Conjecture Generator Comparison

Table 9 compares Claude-4.5-Sonnet against Qwen-QwQ-32B as the conjecture generator, holding all other pipeline stages constant.

Model	Generated	Syntax Valid	Verified	Yield
Claude-4.5-Sonnet	29,330	6,971 (23.8%)	2,608 (8.9%)	8.9%
Qwen-QwQ-32B	29,330	602 (2.1%)	133 (0.5%)	0.5%

Table 9: Conjecture generation comparison. Claude-4.5-Sonnet achieves a yield rate $17.8\times$ higher than Qwen-QwQ-32B. The gap reflects the difficulty of generating Lean4-compatible physics conjectures: physics headers introduce complex namespace and type dependencies that require deep familiarity with the PhysLean codebase.

The full pipeline code will be released to enable the community to reproduce and extend this comparison with future models.

K Math Error Analysis for OOD Generalization

To characterize the nature of the OOD improvement, we manually examined proof attempts on MiniF2F problems where **PhysProver** succeeds but the baseline fails. **PhysProver** produces 3% fewer total proof errors across evaluated trajectories. The reductions concentrate in three error types:

1. `mod_cast` coercion errors: incorrect casting between numeric types ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$) causing type mismatches.

2. Unknown identifier references: calls to deprecated or non-existent Mathlib lemma names.

3. Tactic argument type mismatches: structurally valid tactic calls with incorrect argument types.

We hypothesize that exposure to PhysLean's rich type hierarchy—which requires precise handling of physical quantities across diverse mathematical structures—strengthens the model's semantic understanding of Lean4's type system, yielding these cross-domain benefits.

L Experimental Details

L.1 Prompt Template for Main Experiments

We list the prompt template for DeepSeek-Prover-V2-7B in Figure 7 with a concrete example from **PhysLeanData**. Specifically, in the user round, we append the statement with `sorry` to act as a task for the model. And in the assistant round, we give the model the entire context of the header and the statement. We chose this template because we found it stable for the models to generate proof completions, as it allows the model to directly generate the proof part.

The prompt template for Kimina-Prover and Goedel-Prover is exactly the same except for the special tokens which are shown in Figure 8.

The prompt template for the proprietary models is different due to the inaccessibility of the actual models. Instead, we apply the Chain-of-Thought prompting (Wei et al., 2023), and the template is shown in Figure 9

We also list the prompt template for Claude-4.5-Sonnet to generate conjectures in Figure 10.

L.2 Conjecture Generation

We also list an example of the conjecture, given a header and a lemma, which is shown in Figure 11. In this example, the conjecture is a variant of the original lemma - Both lemmas are basic algebraic properties of matrix-vector multiplication. In other examples we examined, the conjectures consistently fall into predictable categories relative to their preceding lemmas. These variations typically manifest as: (1) special cases obtained by substituting specific values such as zero, (2) definitional unwrapping where a concept is restated in more explicit form, or (3) algebraic variants that apply the same underlying principle to a different argument. Since PhysLean formalizes abstract concepts in mathematical physics, many lemmas establish foundational API properties—such as linearity, group action behavior, or interaction with zero elements—and the conjectures naturally complete this API by covering symmetric or boundary cases. Consequently, these conjectures primarily serve to help models become more familiar with the dataset's conventions, naming patterns, and proof structures to improve the accuracy, rather than to evaluate deep mathematical reasoning capabilities.

Figure 3: **Case 1 — `σ3_selfAdjoint`.** The baseline misapplies `Matrix.ext_iff` as a direct rewrite target; it is a bi-implication (↔), not a rewrite rule for the goal ⊢ $\sigma 3^H = \sigma 3$. **PhysProver** correctly unfolds the definition with `rw [σ3]`, reduces to element-wise equality with `ext i j`, and closes all 2×2 entries by finite case enumeration.

DeepSeek-Prover (FAIL)

```
lemma σ3_selfAdjoint : σ3^H = σ3 := by
  rw [Matrix.ext_iff]
  -- ×Matrix.ext_iff is (↔), not a
  -- direct rewrite for |- σ3^H = σ3
  simp [σ3]
  <;> aesop
-- Error: rewrite failed, did not find
-- instance of pattern in goal |- σ3^H =
    σ3
```

PhysProver (PASS)

```
lemma σ3_selfAdjoint : σ3^H = σ3 := by
  rw [σ3] -- ✓unfolds definition
  ext i j -- ✓element-wise equality
  fin_cases i <;> fin_cases j <;>
    simp [Matrix.transpose_apply,
      conj_re, conj_im, Complex.ext_iff]
  <;> rfl
```

Figure 4: **Case 2 — `getDual?_eq_none_iff_mem_uncontracted`.** The baseline hallucinates `true_and_iff`, a deprecated identifier renamed to `true_and` in the current PhysLean version. **PhysProver** avoids identifier lookup by using `simp [uncontracted] <;> aesop`, letting automation close the goal directly.

DeepSeek-Prover (FAIL)

```
lemma getDual?
    _eq_none_iff_mem_uncontracted
   (i : Fin n) :
   c.getDual? i = none ↔
   i ∈c.uncontracted := by
  constructor
  ·intro h
    simp_all [uncontracted,
      Finset.mem_filter, Finset.mem_univ,
      true_and_iff]
  ·intro h
    simp_all [uncontracted,
      Finset.mem_filter, Finset.mem_univ,
      true_and_iff]
-- Error: unknown identifier '
    true_and_iff'
```

PhysProver (PASS)

```
lemma getDual?
    _eq_none_iff_mem_uncontracted
   (i : Fin n) :
   c.getDual? i = none ↔
   i ∈c.uncontracted := by
  simp [uncontracted]
  <;> aesop
```

Figure 5: **Case 3 — `fderiv_pi'`.** The baseline hallucinates the field access `h.hasFDerivAt.pi`, which does not exist (`HasFDerivAt` has no `.pi` field). **PhysProver** correctly invokes `hasFDerivAt_pi`, the actual Mathlib lemma, demonstrating improved API surface awareness.

DeepSeek-Prover (FAIL)

```
lemma fderiv_pi'
   (h : DifferentiableAt k Φx) :
   fderiv k Φx =
   ContinuousLinearMap.pi fun i =>
     (fderiv k fun x => Φx i) x := by
  apply HasFDerivAt.fderiv
  exact h.hasFDerivAt.pi
-- Error: invalid field 'pi', the
-- environment does not contain
-- 'HasFDerivAt.pi'
```

PhysProver (PASS)

```
lemma fderiv_pi'
   (h : DifferentiableAt k Φx) :
   fderiv k Φx =
   ContinuousLinearMap.pi fun i =>
     (fderiv k fun x => Φx i) x := by
  apply HasFDerivAt.fderiv
  exact hasFDerivAt_pi h
  -- ✓correct Mathlib lemma name
```

Figure 6: An example of the **PhysLeanData**. The black lines denote the header, the blue lines denote the lemma statement, and the red lines denote the proof.

Example of Header, Statement, and the Proof from PhysLean

```
import PhysLean.Mathematics.Fin.Involutions
import PhysLean.QFT.PerturbationTheory.WickContraction.ExtractEquiv
import PhysLean.QFT.PerturbationTheory.WickContraction.Involutions

open FieldSpecification

variable {𝓕 : FieldSpecification}

namespace WickContraction

variable {n : ℕ} (c : WickContraction n)

open PhysLean.List
open FieldStatistic
open Nat

def IsFull : Prop := c.uncontracted = ∅

instance : Decidable (IsFull c) := decEq c.uncontracted ∅

/-- This theorem states that the configuration c is full if and only if, for every index i
    in the finite set of size n, the involution function associated with c does not map i
    to itself.-/

lemma isFull_iff_equivInvolution_no_fixed_point :
IsFull c ↔∀(i : Fin n), (equivInvolution c).1 i ≠i := by
simp only [IsFull, ne_eq]
rw [Finset.eq_empty_iff_forall_notMem]
simp [equivInvolution, toInvolution, uncontracted]
```

Figure 7: Prompt template for DeepSeek Prover with a concrete example

DeepSeek Prover Prompt Template

```
<|begin_of_sentence|> You are an expert in mathematics, physics and Lean 4.
<|User|>Think about and solve the following problem step by step in Lean 4. →Directly give
    the formal statement in Lean 4.

# Problem: This theorem states that the matrix product of the Minkowski metric with itself
    yields the identity matrix.
# Formal statement:
```lean4
import Mathlib.Algebra.Lie.Classical
import Mathlib.Analysis.Normed.Ring.Lemmas

open Matrix

def minkowskiMatrix {d : ℕ} : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ:=
  LieAlgebra.Orthogonal.indefiniteDiagonal (Fin 1) (Fin d) ℝ

namespace minkowskiMatrix

variable {d : ℕ}

scoped[minkowskiMatrix] notation "η" => minkowskiMatrix

/-- This theorem states that the matrix product of the Minkowski metric with itself yields
    the identity matrix.-/
@[simp]
lemma sq : @minkowskiMatrix d * minkowskiMatrix = 1 :=
  sorry
```

<|Assistant|>
```lean4
import Mathlib.Algebra.Lie.Classical
import Mathlib.Analysis.Normed.Ring.Lemmas

open Matrix

def minkowskiMatrix {d : ℕ} : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ:=
  LieAlgebra.Orthogonal.indefiniteDiagonal (Fin 1) (Fin d) ℝ

namespace minkowskiMatrix

variable {d : ℕ}

scoped[minkowskiMatrix] notation "η" => minkowskiMatrix

/-- This theorem states that the matrix product of the Minkowski metric with itself yields
    the identity matrix.-/
@[simp]
lemma sq : @minkowskiMatrix d * minkowskiMatrix = 1 :=
```
```

Figure 8: Prompt template for Kimina Prover and Goedel Prover with a concrete example. They are the same as the Deepseek Prover except for the special tokens, such as the BOS token.

Kimina and Goedel Provers Prompt Template

```
<|im_start|>system
You are an expert in mathematics, physics and Lean 4.<|im_end|>
<|im_start|>user
Think about and solve the following problem step by step in Lean 4. →Directly give the
    formal statement in Lean 4.

# Problem: This theorem states that the matrix product of the Minkowski metric with itself
    yields the identity matrix.
# Formal statement:
```lean4
import Mathlib.Algebra.Lie.Classical
import Mathlib.Analysis.Normed.Ring.Lemmas

open Matrix

def minkowskiMatrix {d : ℕ} : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ:=
  LieAlgebra.Orthogonal.indefiniteDiagonal (Fin 1) (Fin d) ℝ

namespace minkowskiMatrix

variable {d : ℕ}

scoped[minkowskiMatrix] notation "η" => minkowskiMatrix

/-- This theorem states that the matrix product of the Minkowski metric with itself yields
    the identity matrix.-/
@[simp]
lemma sq : @minkowskiMatrix d * minkowskiMatrix = 1 :=
  sorry
```<|im_end|>
<|im_start|>assistant
```lean4
import Mathlib.Algebra.Lie.Classical
import Mathlib.Analysis.Normed.Ring.Lemmas

open Matrix

def minkowskiMatrix {d : ℕ} : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ:=
  LieAlgebra.Orthogonal.indefiniteDiagonal (Fin 1) (Fin d) ℝ

namespace minkowskiMatrix

variable {d : ℕ}

scoped[minkowskiMatrix] notation "η" => minkowskiMatrix

/-- This theorem states that the matrix product of the Minkowski metric with itself yields
    the identity matrix.-/
@[simp]
lemma sq : @minkowskiMatrix d * minkowskiMatrix = 1 :=
```
```

Figure 9: Prompt template for proprietary models for the main experiments.

Proprietary Models Prompt Template

```
Complete the following Lean 4 code:

```lean4
{THE LEAN4 HEADERS AND STATEMENT}
```

Before producing the Lean 4 code to formally prove the given theorem, provide a proof plan
    outlining the main proof steps and strategies.

The plan should highlight key ideas, intermediate lemmas, and proof structures that will
    guide the construction of the final formal proof.
```

Figure 10: Prompt template for Claude-4.5-Sonnet to generate formal conjectures

Claude-4.5-Sonnet Prompt Template for Conjecture Generation

```
You are an expert in mathematics, physics and Lean 4.
You are provided a context, a lemma, and a proof. Your task is to generate a list of 10
    related physics conjecture in formal language based on the context and the seed
    language statements.

The conjectures should be:
1. A meaningful variant of the original theorem: modify hypotheses, generalize structures,
    or extend scope while keeping the core mathematical insight.
2. Must differ significantly in mathematical content (changed assumptions, stronger/weaker
    conclusions, or different algebraic structures) but remain recognizably related.
3. The new conjecture should be in formal language.
4. Do not include the proof.

When generating the conjectures, preserve all specific Lean identifiers exactly as they
    appear in the formal statement. You can also refer to the original formal statement.

Context:
{context}

Natural Language Statement:
{nq}

Original Formal Statement:
{theorem}

Return the final conjectures in JSON format as a dictionary where:
- The key is "conjectures"
- The value is a list of dictionaries
- Each dictionary in the list has a key "statement" whose value is a string containing one
    conjecture

Please read, understand, and then generate a list of conjectures.
```

Figure 11: An example of the conjecture based on the header and the lemma

The Header of the Original Lemma

```
import PhysLean.Relativity.PauliMatrices.SelfAdjoint
import Mathlib.RepresentationTheory.Basic
import PhysLean.Relativity.Lorentz.Group.Basic
import Mathlib.Analysis.InnerProductSpace.PiL2

namespace Lorentz

noncomputable section
open Matrix
open MatrixGroups
open Complex

structure ContrMod (d : ℕ) where
  val : Fin 1 ⊕Fin d →ℝ

namespace ContrMod

variable {d : ℕ}

def toFin1dℝFun : ContrMod d ≃(Fin 1 ⊕Fin d →ℝ) where
  toFun v := v.val
  invFun f := ⟨f⟩
  left_inv _ := rfl
  right_inv _ := rfl

instance : AddCommMonoid (ContrMod d) := Equiv.addCommMonoid toFin1dℝFun

instance : AddCommGroup (ContrMod d) := Equiv.addCommGroup toFin1dℝFun

instance : Module ℝ(ContrMod d) := Equiv.module ℝtoFin1dℝFun

def toFin1dℝEquiv : ContrMod d ≃ₗ[ℝ] (Fin 1 ⊕Fin d →ℝ) :=
  Equiv.linearEquiv ℝtoFin1dℝFun

abbrev toFin1dℝ (ψ : ContrMod d) := toFin1dℝEquiv ψ

@[simps!]
def stdBasis : Basis (Fin 1 ⊕Fin d) ℝ(ContrMod d) := Basis.ofEquivFun toFin1dℝEquiv

abbrev mulVec (M : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ) (v : ContrMod d) :
    ContrMod d := Matrix.toLinAlgEquiv stdBasis M v

scoped[Lorentz] infixr:73 " *ᵥ" => ContrMod.mulVec
```

The Original Lemma

```
/-- This theorem states that for any two real matrices M and N of dimension (1+d) ×(1+d)
     and any vector v in ContrMod d, the matrix-vector product of their difference with v
     equals the difference of their individual matrix-vector products with v.-/
lemma sub_mulVec (M N : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ) (v : ContrMod d) :
    (M - N) *ᵥv = M *ᵥv - N *ᵥv := sorry
```

The New Conjecture

```
lemma mulVec_zero (M : Matrix (Fin 1 ⊕Fin d) (Fin 1 ⊕Fin d) ℝ) : M *ᵥ(0 : ContrMod d) = 0
     := sorry
```