GROUNDFORM: LEARNING TO GROUND AUTOFORMALIZATION WITH VERIFIED HINDSIGHT CREDIT AND GROWING MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoformalization requires a model to reconcile natural-language intent, visual evidence, and domain knowledge in a single precise statement. We present GroundForm, an agent that formalizes both textual mathematics and multimodal physics and learns which knowledge to retrieve along the way. It is trained with Verified Hindsight Credit (VHC): the formalizations that pass verification within a rollout group serve as targets, and a frozen reference model scores how much closer each tool interaction brings the agent to them, so that intermediate retrieval decisions receive their own credit. In addition, GroundForm maintains two verified memories that grow during training. Mathematical experience can be retrieved in both domains, and physical experience supplies laws and conventions missing from the static libraries. On eight benchmarks, GroundForm reaches 80.8% average semantic accuracy in mathematics and 58.2% in physics, 57.6 and 43.6 points above its untrained Qwen3.5-9B backbone. Interventions on tools, memory, and images help locate the source of these gains. VHC increases training time per step by 16.8%, yet it needs 41% fewer GPU-hours to reach the final PhyX-AF accuracy of outcome-only training.

1 INTRODUCTION

Translating an informal problem into a formal statement involves many small semantic decisions: which quantities appear, which assumptions constrain them, and which library objects can express those constraints. A compiler catches ill-typed output, but a statement that compiles may still drop a hypothesis or answer a different question from the one posed. The gap between well-formedness and faithfulness has accordingly been a recurring concern in autoformalization with language models (Wu et al., 2022; Azerbayev et al., 2023a; Zheng et al., 2021).

In multimodal physics these decisions are harder to sidestep. The diagram often fixes an angle, a reference frame, or a circuit topology that the text never states, and the agent must carry this information through a physical law into a formal expression. For the dipole problem in Figure 1, that means combining the torque relation, a norm identity for the cross product, and a conversion from degrees to radians. Finding the right identity is only part of the work; the agent still has to instantiate it with the quantities and conventions of the problem at hand. We refer to this process of linking the input and the agent's reasoning to external knowledge and formal objects as *grounding*.

Retrieval lets an agent draw on knowledge it has not memorized (Lewis et al., 2020; Yao et al., 2022; Schick et al., 2023), but every search is also a decision whose value an endpoint reward cannot isolate. Successful trajectories often include redundant searches, and failed ones sometimes retrieve exactly the right definition. The lack of a canonical gold formalization makes credit assignment harder still, since several Lean statements may express the same problem equally well.

GroundForm approaches this problem from two directions. *Verified Hindsight Credit* (VHC) teaches the agent *how to ground*. For each problem, VHC collects the formalizations accepted within the current rollout group and uses a frozen reference model to measure how readily these targets can be generated from the context at each tool boundary. Changes in this score give a turn-level signal, and a return-to-go propagates later progress back to earlier decisions. *Growing verified memory* extends *what the agent can ground against*: experience from solved problems is abstracted, checked,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

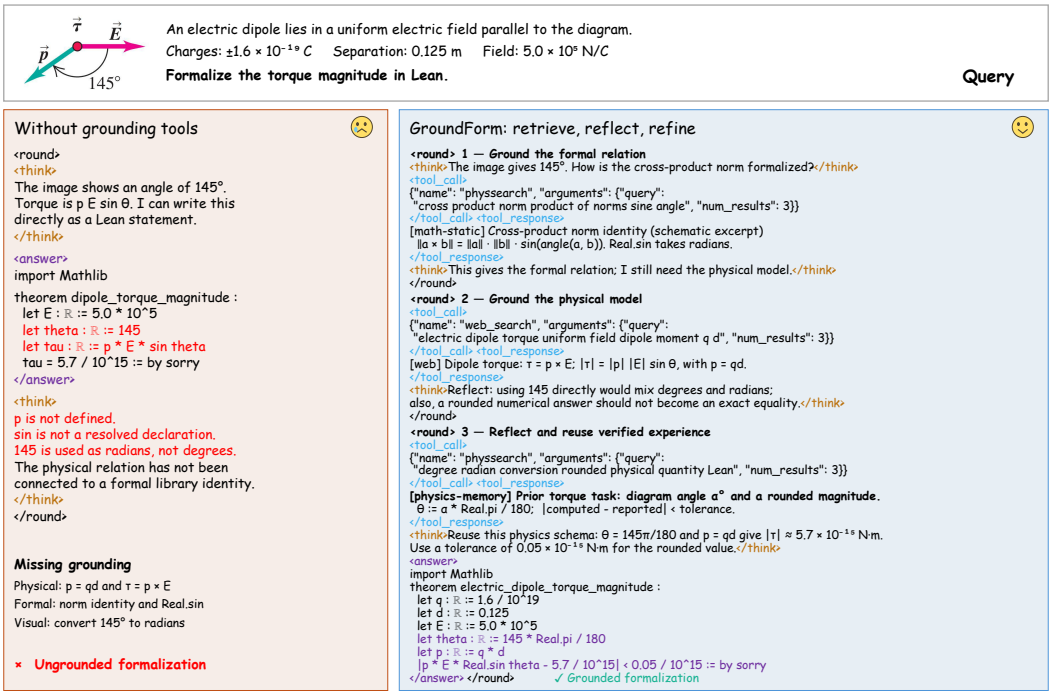


Figure 1: Grounded autoformalization on a dipole problem. The agent relates the diagram to a formal norm identity, retrieves the physical model, and reuses a verified schema for angular units and numerical tolerance. Excerpts are shortened for illustration; the final output is a Lean statement.

and stored for later retrieval. Mathematical entries are available in both domains, while physical entries add laws, reference-frame conventions, and patterns for turning visual information into formal constraints.

Our main contributions are as follows.

- A single grounded autoformalization agent for textual mathematics and multimodal physics that queries MATHSEARCH, PHYSSEARCH, and WEBSEARCH during generation.
- A training scheme that pairs turn-level verified hindsight credit with two growing verified memories, so that the credit signal and the retrievable knowledge improve together.
- Controlled ablations and trajectory analyses of grounding behavior, transfer from mathematics to physics, memory reuse, and training efficiency.

2 RELATED WORK

Autoformalization and multimodal formal reasoning. Neural theorem proving has progressed through proof corpora, curriculum learning, search, and interaction with proof assistants (Polu & Sutskever, 2020; Han et al., 2021; Polu et al., 2022; Lample et al., 2022; Wu et al., 2021; Bansal et al., 2019). Retrieval and symbolic tools link generated proofs to formal libraries (Yang et al., 2023; Jiang et al., 2022a), and informal sketches or repair steps offer additional guidance (Jiang et al., 2022b; First et al., 2023). Large-scale data synthesis and reinforcement learning have produced much stronger provers (Xin et al., 2024a;b; Lin et al., 2025a;b; Wang et al., 2025a), and newer benchmarks broaden the evaluation of formal reasoning (Yu et al., 2025). Closer to our setting is work on statement generation: StepFun-Formalizer combines knowledge with reasoning, ReForm targets semantic consistency through reflective refinement, and MMFormalizer brings autoformalization to multimodal physics (Wu et al., 2025; Chen et al., 2025; Xiong et al., 2026). Benchmarks for visual mathematics and science contain many problems whose constraints can only be read from the diagram (Lu et al., 2022; 2023; Zhang et al., 2024a; Wang et al., 2024b; Yue et al., 2023; Wang et al., 2024a), and geometry systems translate such visual relations into symbolic form (Lu et al., 2021;

Chervonyi et al., 2025; Zhang et al., 2024b). We ask how an agent can learn this translation from tool use and accumulated experience.

Learning reasoning and assigning process credit. Mathematical problem solving with language models has improved through domain pretraining, explicit reasoning traces, and search (Lewkowycz et al., 2022; Azerbayev et al., 2023b; Wei et al., 2022; Wang et al., 2022b; Yao et al., 2023; Zelikman et al., 2022). Reinforcement learning scales outcome-based optimization (Shao et al., 2024; DeepSeek-AI et al., 2025), and outcome and process verifiers provide finer-grained feedback (Cobbe et al., 2021; Lightman et al., 2023). Hindsight methods reuse the outcomes an agent actually achieved to inform its earlier decisions (Andrychowicz et al., 2017; Harutyunyan et al., 2019). VHC adapts this idea to formalization without a canonical answer. Its targets are accepted statements discovered during training, its score comes from a frozen reference model, and its credit is assigned at tool boundaries. The resulting signal is added to the terminal reward in a clipped policy update (Schulman et al., 2017).

Agentic retrieval and experiential memory. Browser agents and tool-using models learn to gather external evidence and act on it (Nakano et al., 2021; Patil et al., 2023; Qin et al., 2023; Liu et al., 2023c). Reflection and iterative revision help models make better use of feedback (Shinn et al., 2023; Madaan et al., 2023), and persistent skill libraries and experience stores support transfer across tasks (Wang et al., 2023; Zhao et al., 2023; Zhu et al., 2023; Park et al., 2023; Packer et al., 2023). Retrieval-augmented generation also benefits from learning when to retrieve and from evidence-driven multimodal interaction (Asai et al., 2023; Chen et al., 2026). We bring these ideas to formalization. In GroundForm, each memory entry records an applicable relation or formal pattern, entries are admitted only after verification, and both domains are served by one retrieval interface, so the knowledge store grows while the policy is learning to use it.

3 METHOD

3.1 TASK AND GROUNDING ENVIRONMENT

Given an input $x = (u, v)$ with text u and an optional image v, the agent produces a Lean 4 statement y. We write a trajectory as $\xi_i = (s_{i,0}, a_{i,1}, o_{i,1}, \ldots, a_{i,K_i}, o_{i,K_i}, y_i)$, where $s_{i,j}$ is the context after tool response $o_{i,j}$ and each action may contain reasoning, a tool choice, and a query. Two checks decide acceptance. The formal check $F(y)$ tests compilation and elaboration, and the semantic judge $C(x, y)$ tests whether the statement keeps the quantities, hypotheses, quantifiers, and target of the input, including conditions that only the image provides. We define

$$B(x, y) = F(y)C(x, y), \qquad \mathrm{Syn} = \mathbb{E}[F(y)], \qquad \mathrm{Sem} = \mathbb{E}[B(x, y)]. \tag{1}$$

Proof bodies are left as `by sorry`, so acceptance depends only on whether the statement is well formed and faithful to the input.

Besides reading the text and image, the agent can call three search tools, each of which takes a text query. MATHSEARCH searches Lean 4 and Mathlib definitions, theorems, and lemmas. PHYSSEARCH searches Physlib declarations, verified physical statements, and a subset of Mathlib. Both local tools return declaration names, signatures, and informal descriptions, which the agent uses to formalize the relations it has identified in the input. WEBSEARCH returns textual explanations from the web. Let L^m, L^p denote static mathematical and physical libraries, and M_t^m, M_t^p the corresponding memories. Let $L_p^m \subseteq L^m$ be the mathematical declarations indexed by PhysSearch. Their local retrieval scopes are

$$\begin{aligned} \textsc{MathSearch}(q) &= \mathcal{R}(q; L^m \cup M_t^m), \\ \textsc{PhysSearch}(q) &= \mathcal{R}(q; L_p^m \cup L^p \cup M_t^m \cup M_t^p). \end{aligned} \tag{2}$$

Every result is tagged with its domain and source, and a physical query may well return a mathematical memory entry. The policy chooses the tool and the query jointly, $\pi_\theta(z, q \mid s)$. The mathematical library follows the formal structure of Mathlib (Community, 2019).

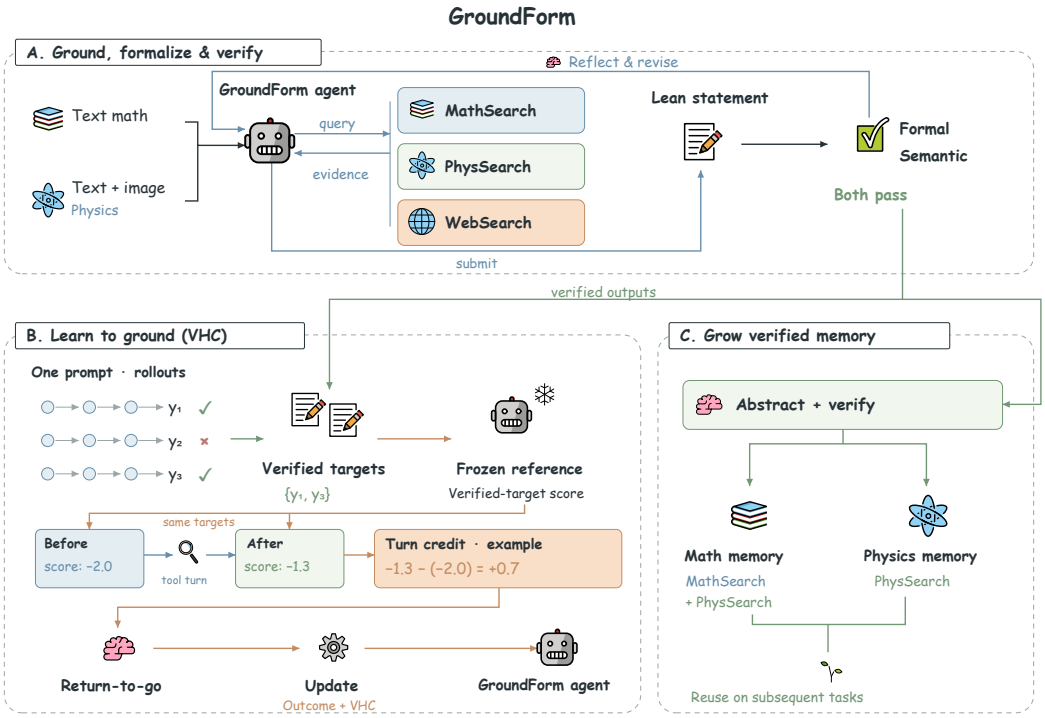


Figure 2: Overview of GroundForm. (A) A search agent queries three tools and writes a Lean statement, which must pass formal and semantic checks. (B) VHC credits each search by how much it brings the agent closer to the accepted outputs. (C) Accepted outputs become memory entries inside the search indexes, so later searches can find them.

3.2 Verified Hindsight Credit

Online targets. For each problem we sample n trajectories and keep the statements that pass both checks:

$$\mathcal{Y}_x^+ = \mathrm{Dedup}\{y_i : B(x, y_i) = 1,\ \xi_i \in \mathcal{G}_x\}. \tag{3}$$

These statements are alternative valid formalizations of the same input. Because the set is shared across the group, a failed trajectory can still receive process credit from targets found by the other trajectories. The set is held fixed while the group is scored, and when it is empty the process term is zero.

Progress at tool boundaries. The SFT model π_{ref} stays frozen throughout reinforcement learning. Teacher forcing gives the length-normalized log-likelihood of each accepted target from a state s:

$$\ell_m(s) = \frac{1}{|y_m^+|} \sum_{r=1}^{|y_m^+|} \log \pi_{\mathrm{ref}}(y_{m,r}^+ \mid s, y_{m,<r}^+), \qquad \Phi_x(s) = \tau \log \sum_{y_m^+ \in \mathcal{Y}_x^+} \exp\left(\frac{\ell_m(s)}{\tau}\right). \tag{4}$$

The potential $\Phi_x(s)$ reflects how easily the accepted targets can be generated from the current context. Length normalization keeps long statements from being penalized simply for their length, and aggregating over the set keeps several formalization paths in play. We use $\tau = 1$ by default.

A turn covers the reasoning and the tool call up to the arrival of the tool response. We define its hindsight reward and return as

$$\Delta_{i,j} = \Phi_x(s_{i,j}) - \Phi_x(s_{i,j-1}), \qquad G_{i,k} = \sum_{j=k}^{K_i} \Delta_{i,j} = \Phi_x(s_{i,K_i}) - \Phi_x(s_{i,k-1}). \tag{5}$$

A positive $\Delta_{i,j}$ means that the interaction moved the context closer to the verified targets. Through the return, an early interaction is also credited for the progress it makes possible later. The return is undiscounted ($\gamma = 1$) and applies to the whole interaction, reasoning and query included.

Policy update. The terminal reward is $R_i = B(x, y_i)$, which we standardize within the rollout group to obtain A_i^{out}. The returns $G_{i,k}$ are normalized separately over the valid tool segments of the same problem, giving $\widetilde{G}_{i,k}$. Policy tokens in tool turn k then receive

$$A_{i,k} = A_i^{\text{out}} + \lambda_V \widetilde{G}_{i,k}, \qquad \lambda_V = 1. \tag{6}$$

This advantage is used in the clipped GRPO objective with KL regularization. Tool responses are observations from the environment and receive no gradient, and tokens after the last tool response use only the outcome advantage. Each trajectory requires $(K_i + 1)|\mathcal{Y}_x^+|$ state–target pairs to be scored, which we evaluate with batched teacher forcing. Appendix A describes the full update and the boundary conventions.

3.3 Growing verified experience

Each memory entry stores a domain, a concept, the conditions under which it applies, reusable formal content, and its provenance. Mathematical entries cover API usage, mappings from informal to formal language, and reusable relations. Physical entries additionally cover laws, reference frames, geometric decompositions, and schemas that link visual, physical, and formal descriptions. In the dipole example of Figure 1, a single earlier entry carries both the angular convention and the numerical tolerance.

Candidate entries are abstracted from accepted trajectories. Qwen3.7-Plus (non-thinking) checks each candidate for semantic consistency and applicability, and any Lean snippet it contains is compiled separately. Candidates are then deduplicated by concept, formal signature, and lexical overlap before admission:

$$M_{t+1}^d = M_t^d \cup \text{Admit}_d(\mathcal{G}_t), \qquad d \in \{m, p\}. \tag{7}$$

Memory is frozen within a rollout group, and new entries become visible to later groups. Standard evaluation uses the final frozen memories. In a separate sequential evaluation, an entry is written only after the current problem has been scored, so it can affect only later problems.

4 Experiments

4.1 Experimental Setup

Datasets. For textual mathematics we use miniF2F (244 problems), ProofNet (186), Putnam (644), and AIME25 (30 problems, three samples each). For physics we use 1,000 problems each from PhyX, SeePhys, and PhysUniBench, together with the 256 problems of PhyX-AF, 97 of which are image-necessary (Shen et al., 2025; Xiang et al., 2025; Wang et al., 2025b; Xiong et al., 2026). The SFT set has 10,073 examples generated with a mixture of Claude Opus 5 and Qwen3.7-Plus, including 2,216 physical trajectories with one image each. The RL set has 185,786 examples, 3,100 of them with images. Appendix B lists the sources and modality counts.

Baselines. We compare against general-purpose language models and specialized autoformalizers, including ReForm-8B; their mathematical results follow the evaluation in ReForm (Chen et al., 2025). All internal variants use the Qwen3.5-9B backbone: the untrained model, SFT, outcome-only GRPO, Shuffled VHC, VHC without growing memory, and the full GroundForm. Shuffled VHC permutes the turn rewards within each trajectory before recomputing the returns.

Evaluation Metrics. Syntactic correctness (syn) is the Lean compilation rate. Semantic consistency (sem), our primary metric, additionally requires the semantic judge to accept the statement (Equation 1). Avg is the unweighted mean of the four dataset scores in a domain. For comparability with ReForm, Table 1 uses Qwen3-235B-A22B-Thinking as the judge; all other experiments use Qwen3.7-Plus non-thinking. Accuracy on the image-necessary subset is reported separately. For calls and tokens per semantic success, we divide the total cost over all evaluation trajectories, failed ones included, by the number of successes.

Table 1: Mathematical results. We report syntactic correctness (syn) and semantic consistency (sem); sem is the primary metric. Best results are in bold, and the best baseline results are underlined.

Model	miniF2F		ProofNet		Putnam		AIME25		Avg	
	syn	sem	syn	sem	syn	sem	syn	sem	syn	sem
Proprietary and Open-source Models										
GPT-5	70.9	66.0	49.5	44.6	61.6	45.8	13.3	13.3	48.8	42.4
Claude-3.7-Sonnet	40.2	34.0	28.5	22.0	20.2	10.1	3.3	3.3	23.1	17.4
Gemini-2.5-Pro	28.7	25.8	23.1	8.1	15.7	6.5	13.3	0.0	20.2	10.1
DeepSeek-R1-0528	38.5	35.2	11.3	9.7	19.7	11.3	16.7	3.3	21.6	14.9
Qwen3-235B-A22B	55.7	43.9	16.7	12.9	33.1	19.9	20.0	13.3	31.4	22.5
Qwen3-32B	57.4	53.3	10.8	8.1	8.2	6.2	10.0	10.0	21.6	19.4
Qwen3-8B	37.7	31.6	7.5	5.9	4.5	3.1	3.3	0.0	13.3	10.2
Autoformalization Models										
DeepSeek-Prover-V1.5-RL	86.1	43.0	36.6	16.1	11.3	8.6	0.0	0.0	33.5	16.9
Goedel-V1-32B-WB	95.1	47.1	48.4	18.3	62.1	9.3	70.0	3.3	68.9	19.5
Goedel-V1-32B-Sonnet	93.4	69.3	47.8	26.3	73.1	17.2	80.0	13.3	73.6	31.5
Kimina-Autoformalizer-7B	92.6	67.6	53.2	23.6	69.7	25.1	80.0	16.7	73.9	33.3
Goedel-Formalizer-V2-8B	97.5	81.1	70.4	47.3	74.5	42.9	66.7	26.7	77.3	49.5
Goedel-Formalizer-V2-32B	97.1	82.0	71.5	50.5	74.2	41.4	66.7	26.7	77.3	50.1
ReForm-8B	98.4	87.7	78.5	65.6	81.9	57.3	**83.3**	46.7	85.5	64.3
Shared-backbone baselines										
Qwen3.5-9B	53.0	40.6	30.1	23.6	31.0	18.6	22.2	10.0	34.1	23.2
SFT	93.0	86.5	65.1	60.2	61.3	48.1	50.0	35.6	67.4	57.6
GRPO	98.0	92.2	76.9	70.4	79.5	65.4	76.7	66.7	82.8	73.7
Shuffled VHC	97.5	91.8	76.3	69.4	79.0	64.6	75.6	65.6	82.1	72.9
VHC	98.4	93.9	80.6	75.3	82.9	70.3	81.1	73.3	85.8	78.2
Ours										
GroundForm	**98.8**	**94.7**	**82.8**	**78.0**	**84.8**	**73.6**	**83.3**	**76.7**	**87.4**	**80.8**

Implementation Details. All RL methods are initialized from the same SFT checkpoint and trained for 160 steps on eight GPUs, with 32 prompts per step, eight rollouts per prompt, a learning rate of 10^{-6}, and a KL coefficient of 0.001. During training, a reply may contain up to 8,192 tokens and ten assistant turns. At evaluation time we sample at temperature 0.6 and allow up to 20,480 reply tokens, nine tool calls, top-5 local retrieval, and 1,536 tokens per tool response. Compilation is checked with the Kimina Lean Server (Santos et al., 2025). Standard evaluation uses the frozen final memory, and sequential writing is evaluated separately.

4.2 Main Results

Mathematical formalization. GroundForm averages 80.8% Sem, 16.5 points above ReForm-8B (Table 1), and improves on every dataset, by 12.4 points on ProofNet and 16.3 on Putnam among others. Among the shared-backbone variants, VHC lifts GRPO from 73.7% to 78.2% and memory brings the average to 80.8%, whereas shuffling the process signal gives only 72.9%. The ranking is the same under the default judge, with GRPO, VHC, and GroundForm at 72.8%, 77.2%, and 79.8% (Appendix E.1).

Physical formalization. GroundForm averages 58.2% Sem (Table 2), against 40.1% for Gemini-2.5-Pro, 49.9% for GRPO, and 54.6% for VHC; its margin over GRPO ranges from 6.8 to 10.2 points across datasets. RL raises Syn over SFT on all four physical datasets, but most of the improvement is semantic. Relative to SFT, GroundForm raises average Syn by 5.9 points (86.3% to 92.2%) and average Sem by 30.1 points (28.1% to 58.2%).

Table 2: Physical results. We report syntactic correctness (syn) and semantic consistency (sem); sem is the primary metric. Best results are in bold, and the best baseline results are underlined.

Model	PhyX		SeePhys		PhysUniBench		PhyX-AF		Avg	
	syn	sem	syn	sem	syn	sem	syn	sem	syn	sem
Proprietary Models										
Gemini-2.5-Pro	73.2	58.1	62.7	43.8	54.7	32.9	53.9	25.4	61.1	40.1
Shared-backbone baselines										
Qwen3.5-9B	86.7	21.1	79.0	13.7	74.7	14.7	85.9	9.0	81.6	14.6
SFT	91.8	39.1	87.2	25.0	80.4	26.9	85.9	21.5	86.3	28.1
GRPO	95.5	71.3	88.1	43.8	82.3	40.1	94.1	44.5	90.0	49.9
Shuffled VHC	95.2	70.5	87.8	43.2	81.9	39.6	93.8	43.4	89.7	49.2
VHC	96.1	75.2	89.6	48.3	84.1	44.6	95.3	50.4	91.3	54.6
Ours										
GroundForm	**96.6**	**78.1**	**90.9**	**51.7**	**85.7**	**48.2**	**95.7**	**54.7**	**92.2**	**58.2**

Table 3: Target-selection ablations for VHC without growing memory, reported as Sem (%). The best result in each row is in bold.

Benchmark	Single accepted target		Full accepted set			
	Random	Most likely	$\tau = 0.5$	$\tau = 1$	$\tau = 2$	$\tau = 4$
Putnam	66.7	67.3	68.3	**69.1**	68.8	67.6
PhyX-AF	47.6	48.3	49.5	**50.4**	50.0	48.8

4.3 Ablation Studies

Hindsight targets. Table 3 compares ways of choosing targets on Putnam and PhyX-AF. Aggregating over the full accepted set works better than using a single target, whether chosen at random or by likelihood, and $\tau = 1$ is best on both benchmarks. The aggregate should therefore keep several accepted formalizations while still favoring those that fit the current context.

Mathematical and physical memory. With the final policy held fixed, each memory improves physical accuracy on its own (Table 4a). Mathematical experience reaches physics problems through the declaration index shared with PhysSearch, but physical experience accounts for the larger gain. Computed before rounding the averages, the gains are 0.55 points for mathematical memory, 2.38 for physical memory, and 3.43 for both together, an interaction of 0.50 points.

Local retrieval. Removing PhysSearch lowers PhyX-AF Sem by 12.5 points and removing MathSearch lowers it by 3.9 points, while removing WebSearch costs only 1.6 points (Table 4b). Access to local formal declarations thus matters most; the visual constraints themselves still come from the image input.

4.4 Credit and Retrieval Behavior

We examine 5,800 tool turns from held-out GRPO trajectories on PhyX-AF and Putnam. From the lowest to the highest quintile of VHC return, the probability of semantic success rises from 0.24 to 0.84, formal reuse from 0.06 to 0.39, and formal dependency from 0.02 to 0.27. VHC has a Spearman correlation of 0.43 with success and an AUROC of 0.74, compared with 0.02 and 0.51 for shuffled credit. These associations are retrospective, but they indicate that the signal follows useful progress in the collected trajectories.

Training with VHC also changes which tools the agent relies on. On PhyX-AF, WebSearch use drops from 96% under GRPO to 36% under VHC and 24% under GroundForm, while MathSearch use increases from 36% to 41% and 47% and the reuse rate of mathematical retrieval results increases from 38.6% to 52.4% and 60.1%. Taken together with the tool interventions, this suggests that the learned policy makes more use of retrieved formal declarations.

Table 4: Memory and tool ablations with the GroundForm policy held fixed. (a) Average physical Sem (%) for different memory contents. (b) PhyX-AF Sem (%) for different tool sets.

(a) Memory	Phys. Avg
Static libraries only	54.8
Mathematical memory only	55.3
Physical memory only	57.1
Both memories	**58.2**

(b) Tools	PhyX-AF
MathSearch + WebSearch	42.2
PhysSearch + WebSearch	50.8
PhysSearch + MathSearch	53.1
All three tools	**54.7**

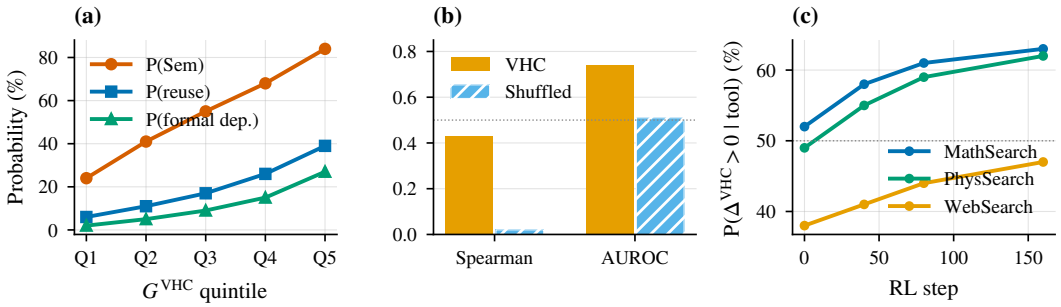


Figure 3: (a–b) Relation of VHC to semantic success and formal reuse. (c) Fraction of turns with positive immediate credit for each tool.

4.5 Memory Reuse and Transfer

How experience is represented matters as well. Mathematical schemas average 198 tokens, against 612 for raw snippets, and reach 79.8% average mathematical Sem. With the policy fixed, growing mathematical memory from zero to 2,410 entries raises Sem from 78.1% to 79.8%. Physical memory reaches 2,930 entries covering 579 concepts, and PhysSearch top-5 coverage increases from 38.2% to 68.3%.

With sequential writing on PhysUniBench and the policy fixed, Sem increases from 48.2% to 48.9%, a net gain of seven successes in one problem order (Appendix C.2).

4.6 Visual Constraints in Formalization

On the image-necessary subset, masking the key region lowers GRPO Sem from 36.1% to 14.4%, and removing the image lowers it to 5.2%; for GroundForm the corresponding figures are 45.4%, 17.5%, and 6.2% (Figure 4a). On the other 159 problems, removing the image changes accuracy by only about two points. Both methods thus depend on the diagram for constraints such as angles, directions, and connections.

We also annotate all 97 image-necessary trajectories of each method by how they use the visual information. GRPO and GroundForm write it into Lean assumptions or the target in 41.2% and 43.3% of trajectories, mention it only in the reasoning in 27.8% each, and ignore it in 30.9% and 28.9%. Semantic success follows the same order. It is highest when the information reaches the Lean statement (67.5% and 81.0%), lower when it stays in the reasoning (25.9% and 33.3%), and close to zero when the image is ignored (3.3% and 3.6%). For both methods, success depends on turning what the image shows into formal constraints. Appendix D gives the counts and the category definitions.

4.7 Efficiency and Evaluation Reliability

VHC raises the time per training step from 1,168.0 to 1,364.2 seconds (16.8%), of which reference scoring accounts for 202 seconds. Even so, it reaches GRPO's final PhyX-AF Sem of 44.5% at step 81 instead of step 160, using 245 rather than 415 GPU-hours, or 41% less. GroundForm gets there by step 59, with 178 GPU-hours and 15,104 trajectories (Figure 5). A more expensive update can therefore still lower the total cost of reaching a given accuracy.

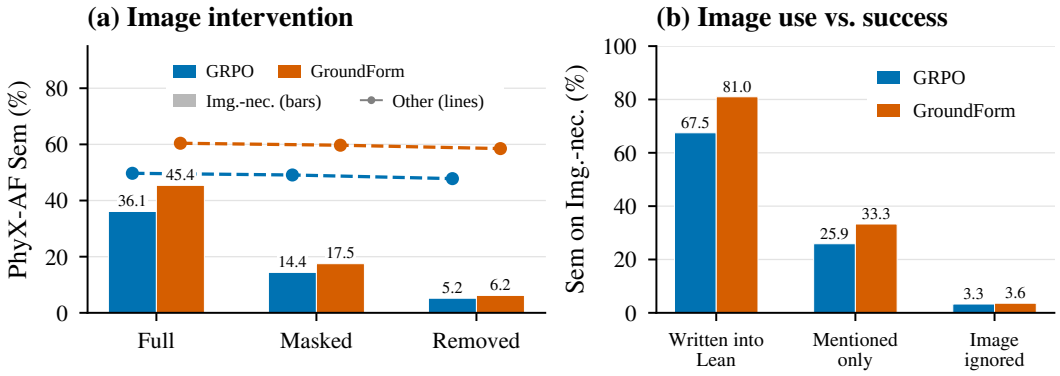


Figure 4: Visual constraints in physical autoformalization. (a) Image interventions on the 97 image-necessary problems (bars) and on the remaining 159 PhyX-AF problems (dashed lines). (b) Semantic success by image-use category over all 97 image-necessary trajectories of each method.

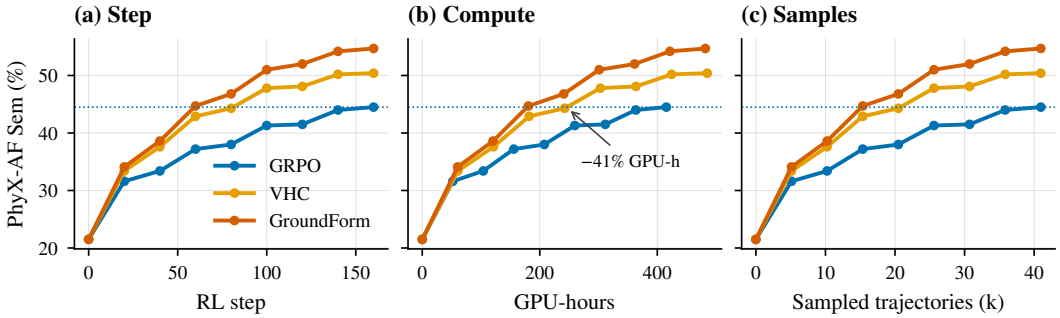


Figure 5: Training efficiency on PhyX-AF. Despite the additional cost of reference scoring, VHC reaches the final semantic accuracy of GRPO with fewer steps, GPU-hours, and sampled trajectories.

The trained policy is also cheaper at inference. In physics, GroundForm makes 36.1% fewer PhysSearch calls and 67.7% fewer WebSearch calls per success than GRPO. Its total falls from 12.04 to 6.40 calls per success (4.42 PhysSearch, 1.21 WebSearch, and 0.77 MathSearch), and generated tokens per success fall from 21.6k to 12.1k, a 44.0% reduction. In mathematics, GRPO, VHC, and GroundForm need 5.56, 4.21, and 3.66 MathSearch calls and 12.1k, 9.6k, and 8.7k tokens per success. Appendix F reports all methods with the same domain-level pooling.

Because semantic evaluation relies on a model judge, we check the judge against human labels and against other judges. Qwen3.7-Plus non-thinking has a reported accuracy of 87.5% on ConsistencyCheck-1K. Replacing the mathematical judge leaves the ranking of methods unchanged: GroundForm obtains 79.8%, 80.8%, 79.0%, and 81.1% under the default judge, Qwen3-235B-Thinking, GPT-5.5, and Claude-Opus-4.6, respectively. Judge agreement and the full results under alternative judges are given in Appendix E.

5 Conclusion

We presented GroundForm, an autoformalization agent for textual mathematics and multimodal physics that links problem constraints to formal declarations through mathematical, physical, and web retrieval. Verified Hindsight Credit trains these retrieval interactions using the formalizations accepted within each rollout group, and two growing memories allow verified experience to be reused across tasks and domains. On eight benchmarks, GroundForm improves semantic consistency, and ablations of target selection, memory, and tools indicate where the improvements come from. Our analyses further relate positive credit to the reuse of retrieved declarations, show that physical formalization succeeds mainly when visual information is written into the formal statement, and show that VHC lowers the compute needed to reach a given accuracy.

AI USE STATEMENT

Generative AI tools assisted with translating and drafting the manuscript, organizing related work, formatting references, checking numerical and cross-table consistency, interpreting reported results, providing feedback on experimental methodology, and preparing figures and LaTeX artifacts. The language models used as policies, semantic judges, reference scorers, and memory-admission models are described in the method and experimental sections. The authors are responsible for the final manuscript and its empirical claims.

REPRODUCIBILITY STATEMENT

Section 3 defines the agent, the hindsight potential, credit assignment, and the memory update. Appendix A gives the training procedure, and Appendix B describes the data and evaluation conventions. Appendices C–F contain per-dataset results, retrieval and memory measurements, judge comparisons, and cost breakdowns. Appendix G provides the full prompt templates and complete example trajectories. The accompanying LaTeX source and figure files reproduce the reported manuscript and plots.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay, 2017. URL https://arxiv.org/abs/1707.01495.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics, 2023a. URL https://arxiv.org/abs/2302.12433.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An Open Language Model For Mathematics, 2023b. URL https://arxiv.org/abs/2310.10631.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond, 2023. URL https://arxiv.org/abs/2308.12966.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025. URL https://arxiv.org/abs/2502.13923.

Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An Environment for Machine Learning of Higher-Order Theorem Proving, 2019. URL https://arxiv.org/abs/1904.03241.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens, 2021. URL https://arxiv.org/abs/2112.04426.

Dongyang Chen, Chaoyang Wang, Dezhao Su, Xi Xiao, Zeyu Zhang, Jing Xiong, Qing Li, Yuzhang Shang, and Shichao Kan. V-Retrver: Evidence-Driven Agentic Reasoning for Universal Multimodal Retrieval, 2026. URL https://arxiv.org/abs/2602.06034.

Guoxin Chen, Jing Wu, Xinjie Chen, Wayne Xin Zhao, Ruihua Song, Chengxi Li, Kai Fan, Dayiheng Liu, and Minpeng Liao. ReForm: Reflective Autoformalization with Prospective Bounded Sequence Optimization, 2025. URL https://arxiv.org/abs/2510.24592.

Yuri Chervonyi, Trieu H. Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Junsu Kim, Vikas Verma, Quoc V. Le, and Thang Luong. Gold-medalist Performance in Solving Olympiad Geometry with AlphaGeometry2, 2025. URL https://arxiv.org/abs/2502.03544.

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, 2021. URL https://arxiv.org/abs/2110.14168.

The mathlib Community. The Lean mathematical library, 2019. URL https://arxiv.org/abs/1910.09336.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2501.12948.

Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-Proof Generation and Repair with Large Language Models, 2023. URL https://arxiv.org/abs/2303.04910.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise Zero-Shot Dense Retrieval without Relevance Labels, 2022. URL https://arxiv.org/abs/2212.10496.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. Proof Artifact Co-training for Theorem Proving with Language Models, 2021. URL https://arxiv.org/abs/2102.06203.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Greg Wayne, Satinder Singh, Doina Precup, and Remi Munos. Hindsight Credit Assignment, 2019. URL https://arxiv.org/abs/1912.02503.

Albert Q. Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. Thor: Wielding Hammers to Integrate Language Models and Automated Theorem Provers, 2022a. URL https://arxiv.org/abs/2205.10893.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs, 2022b. URL https://arxiv.org/abs/2210.12283.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering, 2020. URL https://arxiv.org/abs/2004.04906.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT, 2020. URL https://arxiv.org/abs/2004.12832.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention, 2023. URL https://arxiv.org/abs/2309.06180.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. HyperTree Proof Search for Neural Theorem Proving, 2022. URL https://arxiv.org/abs/2205.11491.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2020. URL https://arxiv.org/abs/2005.11401.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving Quantitative Reasoning Problems with Language Models, 2022. URL https://arxiv.org/abs/2206.14858.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, 2023. URL https://arxiv.org/abs/2301.12597.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, 2023. URL https://arxiv.org/abs/2305.20050.

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, Jiayun Wu, Jiri Gesi, Ximing Lu, David Acuna, Kaiyu Yang, Hongzhou Lin, Yejin Choi, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover-V2: Scaling Formal Theorem Proving with Scaffolded Data Synthesis and Self-Correction, 2025b. URL https://arxiv.org/abs/2508.03613.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning, 2023a. URL https://arxiv.org/abs/2304.08485.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the Middle: How Language Models Use Long Contexts, 2023b. URL https://arxiv.org/abs/2307.03172.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents, 2023c. URL https://arxiv.org/abs/2308.03688.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-GPS: Interpretable Geometry Problem Solving with Formal Language and Symbolic Reasoning, 2021. URL https://arxiv.org/abs/2105.04165.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering, 2022. URL https://arxiv.org/abs/2209.09513.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts, 2023. URL https://arxiv.org/abs/2310.02255.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. AIMO-2 Winning Solution: Building State-of-the-Art Mathematical Reasoning Models with OpenMathReasoning dataset, 2025. URL https://arxiv.org/abs/2504.16891.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. WebGPT: Browser-assisted question-answering with human feedback, 2021. URL https://arxiv.org/abs/2112.09332.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as Operating Systems, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative Agents: Interactive Simulacra of Human Behavior, 2023. URL https://arxiv.org/abs/2304.03442.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large Language Model Connected with Massive APIs, 2023. URL https://arxiv.org/abs/2305.15334.

Stanislas Polu and Ilya Sutskever. Generative Language Modeling for Automated Theorem Proving, 2020. URL https://arxiv.org/abs/2009.03393.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal Mathematics Statement Curriculum Learning, 2022. URL https://arxiv.org/abs/2202.01344.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs, 2023. URL https://arxiv.org/abs/2307.16789.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-Context Retrieval-Augmented Language Models, 2023. URL https://arxiv.org/abs/2302.00083.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction, 2021. URL https://arxiv.org/abs/2112.01488.

Marco Dos Santos, Hugues de Saxcé, Haiming Wang, Ran Wang, Mantas Baksys, Mert Unsal, Junqi Liu, Zhengying Liu, and Jia Li. Kimina Lean Server: A High-Performance Lean Server for Large-Scale Verification, 2025. URL https://arxiv.org/abs/2504.21230.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools, 2023. URL https://arxiv.org/abs/2302.04761.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

Hui Shen, Taiqiang Wu, Qi Han, Yunta Hsieh, Jizhou Wang, Yuyue Zhang, Yuxin Cheng, Zijian Hao, Yuansheng Ni, Xin Wang, Zhongwei Wan, Kai Zhang, Wendong Xu, Jing Xiong, Ping Luo, Wenhu Chen, Chaofan Tao, Zhuoqing Mao, and Ngai Wong. PhyX: Does Your Model Have the "Wits" for Physical Reasoning?, 2025. URL https://arxiv.org/abs/2505.15929.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework, 2024. URL https://arxiv.org/abs/2409.19256.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023. URL https://arxiv.org/abs/2303.11366.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Fei Wang, Xingyu Fu, James Y. Huang, Zekun Li, Qin Liu, Xiaogeng Liu, Mingyu Derek Ma, Nan Xu, Wenxuan Zhou, Kai Zhang, Tianyi Lorena Yan, Wenjie Jacky Mo, Hsiang-Hui Liu, Pan Lu, Chunyuan Li, Chaowei Xiao, Kai-Wei Chang, Dan Roth, Sheng Zhang, Hoifung Poon, and Muhao Chen. MuirBench: A Comprehensive Benchmark for Robust Multi-image Understanding, 2024a. URL https://arxiv.org/abs/2406.09411.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models, 2023. URL https://arxiv.org/abs/2305.16291.

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, Jianqiao Lu, Hugues de Saxcé, Bolton Bailey, Chendong Song, Chenjun Xiao, Dehao Zhang, Ebony Zhang, Frederick Pu, Han Zhu, Jiawei Liu, Jonas Bayer, Julien Michel, Longhui Yu, Léo Dreyfus-Schmidt, Lewis Tunstall, Luigi Pagani, Moreira Machado, Pauline Bourigault, Ran Wang, Stanislas Polu, Thibaut Barroyer, Wen-Ding Li, Yazhe Niu, Yann Fleureau, Yangyang Hu, Zhouliang Yu, Zihan Wang, Zhilin Yang, Zhengying Liu, and Jia Li. Kimina-Prover Preview: Towards Large Formal Reasoning Models with Reinforcement Learning, 2025a. URL https://arxiv.org/abs/2504.11354.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. Measuring Multimodal Mathematical Reasoning with MATH-Vision Dataset, 2024b. URL https://arxiv.org/abs/2402.14804.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text Embeddings by Weakly-Supervised Contrastive Pre-training, 2022a. URL https://arxiv.org/abs/2212.03533.

Lintao Wang, Encheng Su, Jiaqi Liu, Pengze Li, Jiabei Xiao, Wenlong Zhang, Xinnan Dai, Xi Chen, Yuan Meng, Lei Bai, Wanli Ouyang, Shixiang Tang, Aoran Wang, and Xinzhu Ma. PhysUniBench: A Multi-Modal Physics Reasoning Benchmark at Undergraduate Level, 2025b. URL https://arxiv.org/abs/2506.17667.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution, 2024c. URL https://arxiv.org/abs/2409.12191.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models, 2022b. URL https://arxiv.org/abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2022. URL https://arxiv.org/abs/2201.11903.

Minchao Wu, Michael Norrish, Christian Walder, and Amir Dezfouli. TacticZero: Learning to Prove Theorems from Scratch with Deep Reinforcement Learning, 2021. URL https://arxiv.org/abs/2102.09756.

Yuhuai Wu, Albert Q. Jiang, Wenda Li, Markus N. Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with Large Language Models, 2022. URL https://arxiv.org/abs/2205.12615.

Yutong Wu, Di Huang, Ruosi Wan, Yue Peng, Shijie Shang, Chenrui Cao, Lei Qi, Rui Zhang, Zidong Du, Jie Yan, and Xing Hu. StepFun-Formalizer: Unlocking the Autoformalization Potential of LLMs through Knowledge-Reasoning Fusion, 2025. URL https://arxiv.org/abs/2508.04440.

Kun Xiang, Heng Li, Terry Jingchen Zhang, Yinya Huang, Zirong Liu, Peixin Qu, Jixi He, Jiaqi Chen, Yu-Jie Yuan, Jianhua Han, Hang Xu, Hanhui Li, Mrinmaya Sachan, and Xiaodan Liang. SeePhys: Does Seeing Help Thinking? – Benchmarking Vision-Based Physics Reasoning, 2025. URL https://arxiv.org/abs/2505.19099.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. DeepSeek-Prover: Advancing Theorem Proving in LLMs through Large-Scale Synthetic Data, 2024a. URL https://arxiv.org/abs/2405.14333.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search, 2024b. URL https://arxiv.org/abs/2408.08152.

Jing Xiong, Qi Han, Yunta Hsieh, Hui Shen, Huajian Xin, Chaofan Tao, Chenyang Zhao, Hengyuan Zhang, Taiqiang Wu, Zhen Zhang, Haochen Wang, Zhongwei Wan, Lingpeng Kong, and Ngai Wong. MMFormalizer: Multimodal Autoformalization in the Wild, 2026. URL https://arxiv.org/abs/2601.03017.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2022. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, 2023. URL https://arxiv.org/abs/2305.10601.

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.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI, 2023. URL https://arxiv.org/abs/2311.16502.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. MathVerse: Does Your Multi-modal LLM Truly See the Diagrams in Visual Math Problems?, 2024a. URL https://arxiv.org/abs/2403.14624.

Renrui Zhang, Xinyu Wei, Dongzhi Jiang, Ziyu Guo, Shicheng Li, Yichi Zhang, Chengzhuo Tong, Jiaming Liu, Aojun Zhou, Bin Wei, Shanghang Zhang, Peng Gao, Chunyuan Li, and Hongsheng Li. MAVIS: Mathematical Visual Instruction Tuning with an Automatic Data Engine, 2024b. URL https://arxiv.org/abs/2407.08739.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners, 2023. URL https://arxiv.org/abs/2308.10144.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: a cross-system benchmark for formal Olympiad-level mathematics, 2021. URL https://arxiv.org/abs/2109.00110.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, 2023. URL https://arxiv.org/abs/2306.05685.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the Minecraft: Generally Capable Agents for Open-World Environments via Large Language Models with Text-based Knowledge and Memory, 2023. URL https://arxiv.org/abs/2305.17144.

A Training Procedure and Credit Assignment

Algorithm 1 One training iteration of GroundForm

Require: Policy π_θ, frozen reference π_{ref}, memory snapshot (M_t^m, M_t^p), prompt batch
Freeze the memory snapshot for the rollout group.
for each prompt x **do**
  Sample n trajectories; retain tool boundaries and policy-token masks.
  Compile and judge each final statement; compute outcome advantages.
  Construct the deduplicated accepted set $\mathcal{Y}_x^+$.
  if $\mathcal{Y}_x^+ \neq \varnothing$ **then**
    Score all boundary-state–target pairs with π_{ref}.
    Compute potential differences and returns using Eq. (5).
    Normalize returns across valid tool segments for x.
  else
    Set the process contribution to zero.
  end if
  Assign the combined advantage to policy tokens in each tool segment.
end for
Update θ with the clipped objective and KL regularization.
Abstract candidate entries from accepted trajectories.
Check admission with Qwen3.7-Plus non-thinking; compile Lean snippets.
Deduplicate admitted entries and add them to the next memory snapshot.

Token-level optimization. Let t index the generated policy tokens, let $\rho_{i,t}(\theta) = \pi_\theta(a_{i,t} \mid s_{i,t})/\pi_{\text{old}}(a_{i,t} \mid s_{i,t})$, and let $\mathcal{P}_i$ be the set of policy-token positions in trajectory i. The clipped objective is

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{|\mathcal{P}_i|}\sum_{t\in\mathcal{P}_i}\left[\min(\rho_{i,t}A_{i,t}, \text{clip}(\rho_{i,t}, 1-\epsilon_c, 1+\epsilon_c)A_{i,t}) - \beta D_{i,t}\right]\right], \tag{8}$$

where $D_{i,t} = D_{\text{KL}}(\pi_\theta(\cdot \mid s_{i,t}) \| \pi_{\text{ref}}(\cdot \mid s_{i,t}))$. The objective is maximized with the advantages held fixed during the update. For tokens inside a tool segment, $A_{i,t}$ is the advantage of that segment; for final-answer tokens it is the outcome advantage. Tool-response tokens are excluded. Prompts without any accepted output have no process signal, and trajectories without tool calls use only the outcome term. A group in which every trajectory succeeds can still carry a process signal if the returns of its tool turns differ.

Normalization and aggregation. Outcome advantages are $(R_i - \mu_x)/(\sigma_x + \epsilon)$ and process advantages are $(G_{i,k} - \mu_x^G)/(\sigma_x^G + \epsilon)$, where the moments are computed over the valid tool-segment returns of the same prompt and $\epsilon > 0$. If all returns are equal, the process advantage is zero. Every trajectory and boundary state in a group shares the same verified target set. Replacing log-sum-exp with log-mean-exp in Eq. (4) subtracts the constant $\tau \log |\mathcal{Y}_x^+|$ from every potential and thus leaves potential differences and returns unchanged. Because the return is undiscounted, it telescopes to the difference between the potential at the final tool boundary and the potential before the credited turn. The answer-generation segment after the last tool response is trained with the terminal reward.

Shuffled control. We randomly permute $\Delta_{i,j}$ within each trajectory and recompute the returns. The multiset and the sum of turn differences stay the same, but their assignment to interactions changes. Everything else, including the potential scorer and the optimizer, is identical to VHC.

Computational context. Batched state–target scoring faces the same efficiency demands as retrieval-augmented language modeling and long-context generation (Borgeaud et al., 2021; Ram et al., 2023; Liu et al., 2023b). The search indexes could be built with dense retrieval, late-interaction retrieval, or embedding methods (Karpukhin et al., 2020; Khattab & Zaharia, 2020; Santhanam et al., 2021; Wang et al., 2022a; Gao et al., 2022); since VHC defines credit on the returned context, it does not depend on a particular retriever. Techniques for efficient language-model serving and RL systems apply directly to its implementation (Kwon et al., 2023; Sheng et al., 2024).

B Data and Evaluation Protocol

Table 5: Evaluation datasets. Avg uses the four dataset scores in each domain.

Domain	Dataset	Problems	Evaluation unit
Mathematics	miniF2F	244	One problem
	ProofNet	186	One problem
	Putnam	644	One problem
	AIME25	30	Three samples per problem; avg@3
Physics	PhyX	1,000	One problem
	SeePhys	1,000	One problem
	PhysUniBench	1,000	One problem
	PhyX-AF	256	One problem
	PhyX-AF image-necessary	97	Subset, excluded from Avg

B.1 Benchmark Profiles

In all eight evaluations the model must write a Lean 4 statement that keeps the assumptions and target of the problem. Mathematical benchmarks provide only text, while physical benchmarks pair the problem text with a diagram. We score each generated statement for compilation and semantic consistency.

miniF2F. miniF2F is a cross-system formal mathematics benchmark of 488 statements taken from competitions such as AMC, AIME, and IMO and from high-school and undergraduate material (Zheng et al., 2021). Following the autoformalization evaluation in ReForm (Chen et al., 2025), we use its 244-problem test set. The problems call for precise translation of arithmetic, algebraic, and geometric conditions, and each contributes one evaluated statement.

ProofNet. ProofNet consists of 371 undergraduate mathematics examples, each pairing a natural-language theorem and proof with a formal Lean statement (Azerbayev et al., 2023a). The topics come from standard textbooks in real and complex analysis, linear algebra, abstract algebra, and topology. We evaluate on the 186-problem test set with the informal statement as input. Only the statement is generated; doing so correctly requires choosing appropriate types and library objects and keeping every hypothesis within the correct quantifier scope.

Putnam. We use the 644-problem PutnamBench collection from ReForm, which covers college-level competition problems from 1962 to 2023 (Chen et al., 2025). The problems often combine several constraints or mathematical objects. A generated statement counts as correct only if it captures the whole problem, including domain restrictions and the requested conclusion. Each problem contributes one evaluated statement.

AIME25. Following ReForm (Chen et al., 2025), AIME25 contains the 30 problems of the 2025 American Invitational Mathematics Examination. These short competition problems involve arithmetic, algebraic, combinatorial, and geometric conditions. We sample three formalizations per problem and report mean accuracy over all 90 generations (avg@3), which measures the success rate of a single generation.

PhyX. PhyX is a multimodal physical-reasoning benchmark of 3,000 questions spanning mechanics, electromagnetism, thermodynamics, optics, modern physics, and waves and acoustics (Shen et al., 2025), further divided into 25 subdomains and six reasoning types. We evaluate on 1,000 of its problems. Each text–image problem must be translated into a formal statement that relates the given quantities, the relevant physical laws, and the requested target. The diagram often supplies geometry, directions, or experimental structure needed to interpret the text.

SeePhys. SeePhys ranges from middle-school physics to PhD qualifying examinations and covers seven physics domains and 21 diagram categories (Xiang et al., 2025). It focuses on problems that

cannot be solved without the visual information. We evaluate on 1,000 problems. Because the diagrams vary widely, the model has to relate the depicted objects and relations to the quantities and conditions of a Lean statement.

PhysUniBench. PhysUniBench contains 3,304 undergraduate physics questions from eight sub-disciplines, each with one diagram, in open-ended and multiple-choice formats and at five difficulty levels (Wang et al., 2025b). We use 1,000 of its problems both for standard evaluation and for the sequential-memory experiment reported separately. A correct formalization must preserve the physical setup and express how the given conditions determine the target quantity. In the sequential setting, a problem is scored before its experience becomes available to later problems.

PhyX-AF and image necessity. PhyX-AF is a benchmark for multimodal autoformalization, in which visual inputs must be connected to mathematical and physical propositions (Xiong et al., 2026). Our collection has 256 problems. For 97 of them, the text alone lacks conditions needed to specify the problem once the diagram is removed; we call these image-necessary. The other 159 serve as the comparison group in the image-intervention experiment. The image-necessary subset is part of the full PhyX-AF score and is not counted as a fifth dataset in the physical average. Interventions and image-use annotations are reported in Appendix D.

B.2 Training Data

SFT data are generated with a mixture of Claude Opus 5 and Qwen3.7-Plus. The 10,073 examples include mathematical trajectories, physical trajectories with one image each, and Physlib parallel data. The RL set contains 185,786 examples, 3,100 of them with images. Source counts are given in the tables below.

Table 6: SFT data: 10,073 examples. Each of the 2,216 physical trajectories includes one image. Physlib parallel examples pair textual theorem or definition descriptions with formal content and contain no tool interactions.

Source	Examples	Modality
Collected mathematical trajectories	3,857	Text
Collected physical trajectories	2,216	Text + one image
Physlib parallel corpus	4,000	Text, no tools
Total	10,073	2,216 with images

Table 7: RL data: 185,786 examples, including 3,100 with images.

Source	Examples	Modality
Lean-Workbook	89,219	Text mathematics
AoPS	30,000	Text mathematics
NuminaMath-1.5	25,510	Text mathematics
DAPO-Math-17k	17,917	Text mathematics
PhysicsEval (train + test)	19,603	Text physics
PhyX full test	3,000	Text + image
Physlib entries collected from evaluation problems	437	Text
PhyX-100	100	Text + image
Total	185,786	3,100 with images

Standard evaluation uses the memory as it stands at the end of training. In the online PhysUniBench experiment, memory is written only after a problem has been scored, and completed problems are never rescored.

Acceptance and cost. Syn is the compilation rate; Sem additionally requires the semantic judge to accept the statement. The domain average is $\frac{1}{4}\sum_{d=1}^{4}\mathrm{Sem}_d$, and likewise for Syn. For cost, let $k_{m,d}$

denote the number of semantic successes of method m on dataset d and $C_{m,d,T}$ the total number of calls to tool T, failed trajectories included. The pooled cost is

$$c_{m,T} = \frac{\sum_d C_{m,d,T}}{\sum_d k_{m,d}} = \frac{\sum_d k_{m,d}\, c_{m,d,T}}{\sum_d k_{m,d}}. \tag{9}$$

Tokens per success are computed over the same trajectories. Since failed trajectories add to the numerator, these costs can exceed the budget of a single problem. We count assistant turns as dialogue turns and tool calls as individual events, so one turn may contain several calls.

Retrieval measurements. Concept reuse indicates whether a nontrivial retrieved formal identifier (a definition, type, theorem, or instance of at least four characters) appears in the final Lean statement. Source share@5 is computed over returned items and hit rates over queries. For the source-conditioned semantic results, we split the tasks that use the tool into hit and non-hit groups, using mathematical tasks for MathSearch and physical tasks for PhysSearch. Each of these statistics has its own unit of observation.

Multimodal and inference context. Reasoning over images builds on visual instruction tuning and vision–language connectors (Liu et al., 2023a; Li et al., 2023), and the Qwen-VL, Qwen2-VL, and Qwen2.5-VL models extend perception and grounding to varied resolutions and input types (Bai et al., 2023; Wang et al., 2024c; Bai et al., 2025). Work on test-time scaling and mathematical sampling shows that results depend on the inference budget (Snell et al., 2024; Muennighoff et al., 2025; Moshkov et al., 2025); we therefore keep the budget fixed across all internal methods.

C Retrieval and Memory Analysis

C.1 Tool Availability

We remove one or more search tools at inference time while keeping the policy fixed. The per-dataset scores below show which retrieval sources physical formalization relies on. PS, MS, and WS denote PhysSearch, MathSearch, and WebSearch.

Table 8: Physical Sem (%) after tool removal from the fixed GroundForm policy. PS: PhysSearch; MS: MathSearch; WS: WebSearch.

Available tools	PhyX	SeePhys	PhysUniBench	PhyX-AF
PS + MS + WS	**78.1**	**51.7**	**48.2**	**54.7**
PS + MS	77.2	50.3	47.0	53.1
PS + WS	75.6	48.5	45.1	50.8
PS only	73.9	46.8	43.4	48.4
MS + WS	68.7	41.0	37.7	42.2

C.2 Memory Content and Transfer

Table 9: Physical Sem (%) with different memory contents. The GroundForm policy is fixed across memory variants; VHC is a separately trained reference. Sequential writing updates memory only on PhysUniBench.

Configuration	PhyX	SeePhys	PhysUniBench	PhyX-AF	Avg
VHC (reference)	75.2	48.3	44.6	50.4	54.6
Static-only	75.4	48.5	44.7	50.4	54.8
Mathematical memory only	75.8	48.9	45.3	51.2	55.3
Physical memory only	77.3	50.6	47.1	53.5	57.1
Step-80 memory snapshot	77.2	50.7	47.3	53.3	57.1
Both memories	78.1	51.7	48.2	54.7	58.2
Sequential writing	78.1	51.7	48.9	54.7	58.4

Only the PhysUniBench score differs in the sequential row. Computed before the dataset averages are rounded, mathematical memory adds 0.55 points, physical memory 2.38, and both together 3.43, for an interaction of 0.50 points.

Table 10: Memory representations. Sem (%) is the four-dataset mean under the default judge; Tokens/item is mean entry length. Physical extensions build on the abstracted schema. A dash denotes an unreported value.

Representation	Math Sem	Physics Sem	Tokens/item
Static-only	78.1	54.8	–
Raw successful snippet	78.6	55.2	612
Verified formal artifact	79.3	56.3	284
Verified abstracted schema	79.8	56.9	198
+ Physical relation	–	57.3	176
+ Physical-to-formal bridge	–	57.8	231
+ Multimodal schema	–	58.2	247

C.3 Retrieval Sources and Memory Growth

Figure 6 shows what the tools return, and Figure 7 tracks how much physical experience is available and used during training.

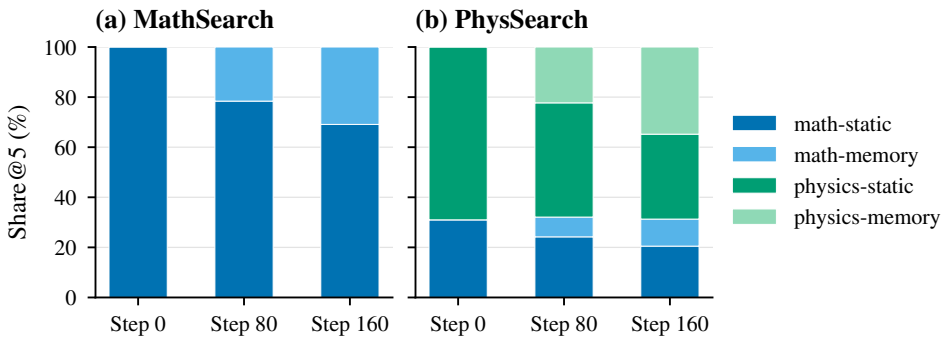


Figure 6: Composition of retrieval sources. Mathematical experience can be retrieved through both MathSearch and PhysSearch. Share@5 is computed over returned items.

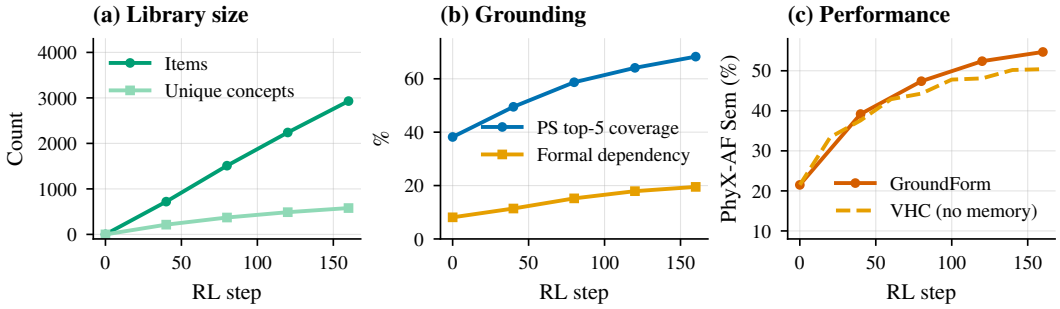


Figure 7: Growth of the physical experience library: entries and concepts, retrieval coverage and downstream dependency, and PhyX-AF semantic accuracy.

Table 11: Sequential PhysUniBench evaluation in one problem order. Each window contains 500 problems; Sem, Physics hit, and Web use are percentages. Latency is seconds per problem.

Window	Successes	Window Sem	Cumulative Sem	Physics hit	Web use	Latency (s)
1–500	242	48.4	48.4	41.8	42	71
501–1,000	247	49.4	48.9	44.0	39	68

D Image Interventions and Visual Constraints

We evaluate the same PhyX-AF problems with the full image, with the key region masked, and with text only, reporting the 97 image-necessary problems separately from the other 159. Figure 4 shows semantic consistency on both subsets. On the other subset, removing the image lowers accuracy by about two points for each method.

For each method, annotators assign each of the 97 image-necessary trajectories to one of three mutually exclusive categories. A trajectory is *written into Lean* if information from the image appears in a Lean assumption or in the target, *mentioned only* if the reasoning discusses this information but the final statement does not express it, and *image ignored* if the image information is not used at all. Category shares are computed over all 97 trajectories and category Sem over the trajectories within each category. Table 12 lists the counts behind Figure 4b.

Table 12: Image-use categories and semantic success on the image-necessary subset. Each method contributes 97 trajectories.

Image use	Trajectories	Share (%)	Successes	Sem (%)
GRPO				
Written into Lean	40	41.2	27	67.5
Mentioned only	27	27.8	7	25.9
Image ignored	30	30.9	1	3.3
Total	97	100	35	36.1
GroundForm				
Written into Lean	42	43.3	34	81.0
Mentioned only	27	27.8	9	33.3
Image ignored	28	28.9	1	3.6
Total	97	100	44	45.4

For both methods, trajectories that ignore the image almost never succeed, and mentioning the image only in the reasoning is usually not enough. Most successes come from trajectories that write the visual information into formal constraints. The two methods distribute their trajectories over the categories in similar proportions.

E Judge Validation and Human Evaluation

As is common for model-based evaluation, we measure how well the judge agrees with human labels (Zheng et al., 2023). ConsistencyCheck-1K combines the 859 expert-labeled examples from ReForm with 141 new items: 71 Putnam examples, 30 AIME25 examples, and 40 hard negative outputs. Two annotators labeled the new items independently ($\kappa = 0.82$), and a third resolved disagreements. Positive examples make up 63.8% of the collection. All judges use the same prompt and greedy decoding.

Table 13: Reported judge accuracy on ConsistencyCheck-1K (1,000 examples). Accuracy is a percentage; cost is relative to Qwen3.7-Plus non-thinking.

Judge	Accuracy	Relative cost
GPT-5	82.1	4.1
Gemini-2.5-Pro	85.3	3.6
Claude-3.7-Sonnet	77.6	3.2
DeepSeek-R1-0528	78.5	1.9
Qwen3-235B-A22B-Thinking	82.4	1.7
QwQ-32B	78.3	1.2
CriticLean-14B	78.6	0.3
GPT-5.5	88.5	4.6
Claude-Opus-4.6	87.6	5.3
Qwen3.7-Plus (thinking)	88.1	1.6
Qwen3.7-Plus (non-thinking)	87.5	1.0

E.1 Results under the Default Semantic Judge

Table 14: Mathematical Syn / Sem (%) under the default Qwen3.7-Plus non-thinking judge.

Method	miniF2F	ProofNet	Putnam	AIME25	Avg
SFT	93.0/86.1	65.1/59.1	61.3/47.0	50.0/34.4	67.4/56.7
GRPO	98.0/91.8	76.9/69.4	79.5/64.3	76.7/65.6	82.8/72.8
Shuffled VHC	97.5/91.4	76.3/68.3	79.0/63.5	75.6/64.4	82.1/71.9
VHC	98.4/93.4	80.6/74.2	82.9/69.1	81.1/72.2	85.8/77.2
GroundForm	98.8/94.3	82.8/76.9	84.8/72.4	83.3/75.6	87.4/79.8

Table 15: Mathematical Sem (%), macro-averaged over four datasets under each judge. All columns evaluate the same method outputs.

Method	Qwen3.7-Plus	Qwen3-235B	GPT-5.5	Claude-Opus-4.6
SFT	56.7	57.6	56.0	57.9
ReForm-8B	63.6	64.3	62.9	64.9
Shuffled VHC	71.9	72.9	71.2	73.3
GRPO	72.8	73.7	72.0	74.1
VHC	77.2	78.2	76.5	78.6
GroundForm	79.8	80.8	79.0	81.1

E.2 Failure Analysis

We sample 200 failures from each comparison group and sort them into error categories to identify recurring failure modes.

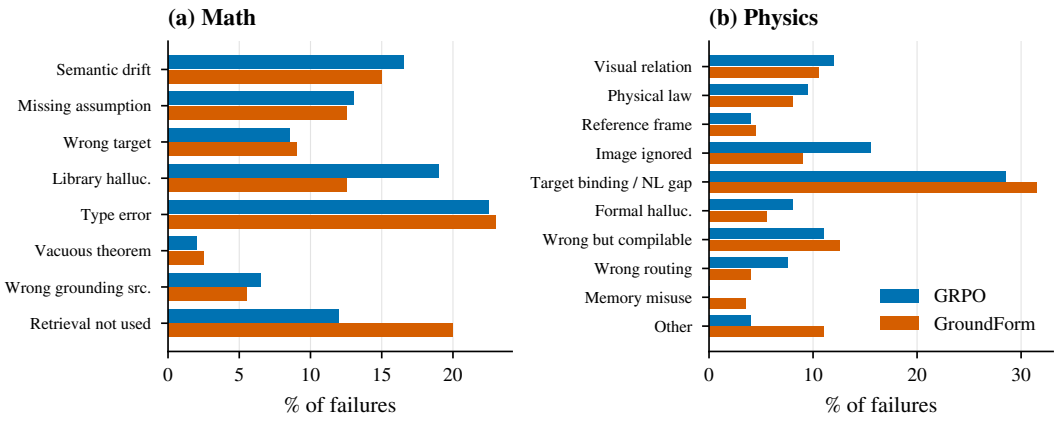


Figure 8: Error categories in 200 sampled failures per comparison group. The values give the composition of failures and should not be read as error rates on the full evaluation set.

F Training and Inference Costs

Table 16: Training time per step in seconds on eight GPUs.

Component	GRPO	VHC	GroundForm
Rollout, tools, verification	911.1	905.3	862.0
Old/reference log probabilities	46.6	46.6	46.6
VHC scoring	–	202.0	198.0
Policy update	201.2	201.2	201.2
Asynchronous memory builder (blocking)	–	–	38.0
Other	9.1	9.1	9.1
Total	1,168.0	1,364.2	1,354.9

Of the 202 seconds spent on VHC, boundary-state extraction takes five, reference prefill 121, target scoring 67, potential aggregation six, and return computation three.

Table 17: Pooled tool calls and generated tokens per semantic success, separately for physics and mathematics. All evaluation trajectories contribute to the cost numerator.

Method	Physics					Mathematics		
	PS	WS	MS	Total	Tokens	MS	WS	Tokens
SFT	5.74	1.51	0.67	7.92	10.9k	5.81	0.11	8.3k
GRPO	6.92	3.75	1.37	12.04	21.6k	5.56	0.88	12.1k
Shuffled VHC	7.06	3.86	1.40	12.32	22.2k	5.68	0.92	12.4k
VHC	5.35	1.82	0.83	8.00	14.3k	4.21	0.49	9.6k
GroundForm	4.42	1.21	0.77	6.40	12.1k	3.66	0.35	8.7k

In Table 17, total costs and semantic successes are pooled within each domain before dividing.

G KEY PROMPTS AND WORKED EXAMPLES

This section reproduces the exact prompt templates used for generation and semantic judging, followed by three complete assistant trajectories, one from ProofNet and two from PhyX. Each trace is split into consecutive panels that read from top to bottom. For physical problems, the image is passed as separate vision content.

G.1 PROMPT TEMPLATES

Student task prompt

Shared evaluation prefix · literal template

```
<|im_start|>user
Think step by step to translate the mathematical problem in natural language to Lean 4, and verify the consistency.
{informal_statement}<|im_end|>
<|im_start|>assistant
```

Semantic judge · user prompt

First consistency check · literal template

```
The Natural Language Statement: {informal_statement}

The Formal Statement in Lean4:
```lean4
{formal_statement}
```

Previous autoformalization and consistency Comments: (None - first check.)

Think about the consistent result:
```

Figure 9: Student task prompt shared by all evaluations (top) and the semantic-judge user template (bottom).

Semantic judge · system prompt

Qwen3.7-Plus · non-thinking · evaluated after generation

Your role is a Lean4 expert. Please help me check consistency between natural language expression and its Lean4 formal statement.

 # Guidelines for Consistency Check

1. Core Checking Requirements:
   - Carefully compare the Natural Language Statement and the Formal Statement in Lean4 through a rigorous and explicit process.
   - Determine if the Lean theorem statement is an exact and faithful formalization of the mathematical problem.
   - Physics "find X" problems: reject generic textbook identities that omit the problem's given data / asked quantity; accept concrete encodings that determine X from the given setup (ℝ/Real is fine when Physlib names are sparse).
   - Reject vacuous statements such as `∃ x, True`, `∃ x, x = x`, or conclusions that do not constrain the asked quantity.
   - If the result is Incorrect, briefly list all inconsistencies and reasons in your analysis.

 # Evaluation Stages (all required)

Stage 1: Restate the NL objects, hypotheses, and the asked conclusion.
Stage 2: Restate the Lean binders, types, hypotheses, and conclusion.
Stage 3: Check domains, boundaries, quantifiers, and logical connectives.
Stage 4: Check that the Lean conclusion is exactly the asked claim, not a weaker, stronger, or vacuous substitute.
Stage 5: Give one final verdict.

 # Output Format

You MUST end with:
<consistency>Correct</consistency>
or
<consistency>Incorrect</consistency>
<comments>
brief reasons
</comments>

Figure 10: Semantic-judge system prompt (Qwen3.7-Plus, non-thinking), applied after generation.

Tool interaction format

Runtime reference · not part of the evaluation prompt

<tool_call>
{"name": "leansearch", "arguments": {"query": "...", "num_results": 8}}
</tool_call>

<tool_response>
...
</tool_response>

Figure 11: Tool-call and tool-response format used at runtime. It is not part of the evaluation prompt.

G.2 Worked Example Prompts

Figure 12 shows the user prompts of the three examples; their full assistant traces follow in three panels each.

ProofNet · user prompt

Mathematics · exact user text

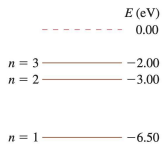
Think step by step to translate the mathematical problem in natural language to Lean 4, and verify the consistency.
For x an element in G show that x and x^{-1} have the same order.

PhyX · user prompt

Text followed by the supplied image · no image caption in prompt

Think step by step to translate the mathematical problem in natural language to Lean 4, and verify the consistency.
What is the ionization energy of element X?

The first three energy levels of the fictitious element X are shown in figure.



Input image

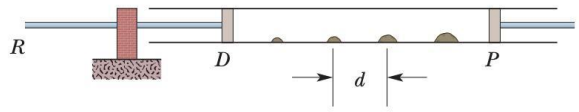
Separate vision content

PhyX · user prompt

Text followed by the supplied image · no image caption in prompt

Think step by step to translate the mathematical problem in natural language to Lean 4, and verify the consistency.
If $f = 4.46 \times 10^3$ Hz and the separation between ridges is 9.20 cm, what is the speed of sound in the gas?

In figure, a rod R is clamped at its center; a disk D at its end projects into a glass tube that has cork filings spread over its interior. A plunger P is provided at the other end of the tube, and the tube is filled with a gas. The rod is made to oscillate longitudinally at frequency f to produce sound waves inside the gas, and the location of the plunger is adjusted until a standing sound wave pattern is set up inside the tube. Once the standing wave is set up, the motion of the gas molecules causes the cork filings to collect in a pattern of ridges at the displacement nodes.



Input image

Separate vision content

Figure 12: User prompts for the worked examples: ProofNet order of an inverse (top), PhyX atomic energy levels (middle), and PhyX standing sound wave (bottom). The diagrams are supplied as separate vision content.

G.3 MATHEMATICS: ORDER OF AN INVERSE (PROOFNET)

ProofNet · full assistant trace | panel 1/3

Same figure, continued in source order. See the prompt figure for input.

1 / 3 · Continue top to bottom

<think>
 # Round 1 (Autoformalization)

<round>

<tool_call>
{"name": "leansearch", "arguments": {"query": "group element order inverse same order", "num_results": 8}}
</tool_call>
<tool_response>
[1] ['orderOf_inv'] (theorem)
(x : G) : orderOf x⁻¹ = orderOf x
For any element x in a group G, the order of its inverse x^{-1} is equal to the order of x, i.e., $\operatorname{ord}(x^{-1}) = \operatorname{ord}(x)$.
[2] ['isOfFinOrder_inv_iff'] (theorem)
{x : G} : IsOfFinOrder x⁻¹ ↔ IsOfFinOrder x
For any element x in a group G, x has finite order if and only if its inverse x^{-1} has finite order. This means there exists a positive integer n such that $x^n = 1$ if and only if there exists a positive integer m such that $(x^{-1})^m = 1$, where 1 denotes the identity element of G.
[3] ['IsOfFinOrder', 'of_inv'] (theorem)
∀ {G : Type u_1} [inst : Group G] {x : G}, IsOfFinOrder x⁻¹ → IsOfFinOrder x
If the inverse of an element $ x $ in a group $ G $ has finite order, then $ x $ itself has finite order.
[4] ['IsOfFinOrder', 'inv'] (theorem)
∀ {G : Type u_1} [inst : Group G] {x : G}, IsOfFinOrder x → IsOfFinOrder x⁻¹
If an element $ x $ of a group $ G $ has finite order, then its inverse $ x^{-1} $ also has finite order.
[5] ['addOrderOf_neg'] (theorem)
∀ {G : Type u_1} [inst : AddGroup G] (x : G), addOrderOf (-x) = addOrderOf x
For any element $ x $ in an additive group $ G $, the additive order of $ -x $ is equal to the additive order of $ x $.

Figure 13: Full assistant trace for the ProofNet example (panel 1 of 3).

ProofNet · full assistant trace | panel 2/3

Same figure, continued in source order. See the prompt figure for input.

2 / 3 · Continue top to bottom

[6] ['inv_eq_self_of_orderOf_eq_two'] (theorem)
{x : G} (hx : orderOf x = 2) : $x^{-1} = x$
For any element x in a group G such that the order of x is 2 (i.e., $x^2 = 1$), it holds that the inverse of x is equal to x itself, i.e., $x^{-1} = x$.
[7] ['orderOf_units'] (theorem)
{y : G^{x}} : orderOf (y : G) = orderOf y
For any invertible element $ y $ in a monoid $ G $, the order of $ y $ as an element of $ G $ is equal to its order as an element of the group of units $ G^□imes $.
[8] ['SemiconjBy', 'orderOf_eq'] (theorem)
[Group G] (a : G) {x y : G} (h : SemiconjBy a x y) : orderOf x = orderOf y
In a group G, if an element a semiconjugates x to y (i.e., $a x = y a$), then the order of x is equal to the order of y. This means that semiconjugacy relations preserve the order of group elements.
</tool_response>
Now I have the relevant Lean 4 statements. The key result is `orderOf_inv` which states exactly what we need: for any element x in a group G, the order of its inverse equals the order of x.

Let me formalize this statement:

```lean4
import Mathlib

/-- For any element x in a group G, x and $x^{-1}$ have the same order. -/
theorem order_of_inverse_eq_order_of_element (G : Type*) [Group G] (x : G) :
  orderOf $x^{-1}$ = orderOf x := by sorry
```

 # Round 1 (Semantic Self-validation)

Figure 14: Full assistant trace for the ProofNet example (panel 2 of 3).

ProofNet · full assistant trace | panel 3/3

Same figure, continued in source order. See the prompt figure for input.

3 / 3 · Continue top to bottom

Your brief analysis:
1. Math Assertion Analysis: The problem states that for any element x in a group G, x and its inverse x^{-1} have the same order. Key components: group G, element $x \in G$, order of x equals order of x^{-1}.
2. Lean Statement Analysis (Proof Ignored): The theorem defines G as a Group, x as an element of G, and asserts orderOf x^{-1} = orderOf x. Uses Group instance, orderOf function, and inverse notation.
3. Comparative Verification:
  - "Same order" is correctly formalized as equality of orderOf values.
  - Group structure is properly captured via [Group G].
  - Variables and quantifiers align (x is arbitrary in G).
  - No missing elements or semantic drift.
4. Conclusion: The Lean statement correctly formalizes the problem.
5. Accuracy Confirmation: All components match perfectly. The theorem directly states the required equality of orders for x and its inverse in a group.

Result: Correct

Therefore, the outcome of the consistency verification is: Correct

</round>
</think>

<answer>

```lean4
import Mathlib

/-- For any element x in a group G, x and $x^{-1}$ have the same order. -/
theorem order_of_inverse_eq_order_of_element (G : Type*) [Group G] (x : G) :
  orderOf $x^{-1}$ = orderOf x := by sorry
```

</answer>

Figure 15: Full assistant trace for the ProofNet example (panel 3 of 3).

G.4 PHYSICS: ATOMIC ENERGY LEVELS (PHYX)

PhyX · full assistant trace | panel 1/3

Same figure, continued in source order. See the prompt figure for input.

1 / 3 · Continue top to bottom

 # Round 1 (Autoformalization)

<round>

<tool_call>
{"name": "physsearch", "arguments": {"query": "ionization energy atomic energy levels electron volts", "num_results": 8}}
</tool_call>
<tool_response>
[1] DimEnergy.electronVolt (definition) [physlib]
: DimEnergy
Energy is represented as a dimensional quantity corresponding to 1 electron volt, which is equal to approximately 1.602176634 \times 10^{-19} \text{ J}.
[2] QuantumMechanics.HydrogenAtom.potential_eq (theorem) [physlib]
: H.potential = fun x ↦ -H.k * ‖x‖⁻¹
The potential energy of the hydrogen atom, denoted as $H.potential$, is defined as $H.potential = -H.k \cdot \|x\|^{-1}$, where $H.k$ represents a constant related to the hydrogen atom and $\|x\|$ is the distance from the nucleus.
[3] DimEnergy.calorie (definition) [physlib]
: DimEnergy
The dimensional energy corresponding to 1 calorie is represented as 4.184 joules.
[4] ChargeUnit.elementaryCharge (definition) [physlib]
: ChargeUnit
The charge unit of elementary charge is defined as \(1.602176634 \times 10^{-19}\) coulombs.
[5] QuantumMechanics.HydrogenAtom.coulomb_potential (theorem) [physlib]
∀ (self : QuantumMechanics.HydrogenAtom), self.potential = fun x => -self.k * ‖x‖⁻¹
For any hydrogen atom represented as a quantum system, the Coulomb potential is defined as \(V(x) = -k \cdot \|x\|^{-1} \), where \(k \) is a constant associated with the hydrogen atom and \(x \) represents the position vector.
[6] Combinatorics.Additive.termE[_,_] (definition) [mathlib]
: Lean.ParserDescr
The notation $E[A, B]$ represents the additive energy between two sets A and B, which counts the number of quadruples $(a, b, c, d) \in A \times B \times A \times B$ such that $a + b = c + d$.
[7] Mathlib.Tactic.BicategoryLike.Atom₁.e (definition) [mathlib]
Mathlib.Tactic.BicategoryLike.Atom₁ → Lean.Expr
The function $ e $ maps elements of the inductive type for bicategory-like atomic components to Lean expressions, converting atomic elements into their syntactic representation in the Lean theorem prover.
[8] Mathlib.Tactic.Widget.StringDiagram.AtomNode.vPos (definition) [mathlib]
Mathlib.Tactic.Widget.StringDiagram.AtomNode → ℕ
A function that assigns a natural number representing the vertical position to each atomic node in a string diagram.
</tool_response>

<tool_call>
{"name": "web_search", "arguments": {"query": "ionization energy definition atomic physics energy levels eV", "num_results": 5}}
</tool_call>
<tool_response>

Figure 16: Full assistant trace for the atomic energy-level example (panel 1 of 3).

PhyX · full assistant trace | panel 2/3

Same figure, continued in source order. See the prompt figure for input.

2 / 3 · Continue top to bottom

[1] Ionization Energy Confusion
In my Barron's SAT Physics book, it states that ionization energy is equal to the absolute value of the ground state energy. This doesn't make sense ...
https://physics.stackexchange.com/questions/416398/ionization-energy-confusion
[2] Ionization energy
In physics, ionization energy (IE) is usually expressed in electronvolts (eV) or joules (J). In chemistry, it is expressed as the energy to ionize a mole of ...
https://en.wikipedia.org/wiki/Ionization_energy
[3] Ionization Energy - Basic Introduction
This chemistry video tutorial provides a basic introduction into Ionization Energy. It discusses the periodic trends and exceptions as well ...
https://www.youtube.com/watch?v=LNVEBURpTcQ
[4] Ionization energy trends (video)
Ionization is the process of removing an electron from a neutral atom (or compound). The energy required to remove an electron is the ionization energy.
https://www.khanacademy.org/science/hs-chemistry/x2613d8165d88df5e:atomic-models-and-periodicity/x2613d8165d88df5e:periodic-trends/v/ionization-energy-trends-hs
[5] Ionization Energy - an overview
Ionization energy is defined as the energy required to remove an electron from a neutral atom or molecule, resulting in the formation of a positively charged ...
https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/ionization-energy
</tool_response>

Based on the problem description and image:
- The energy levels are: E_1 = -6.50 eV (n=1), E_2 = -3.00 eV (n=2), E_3 = -2.00 eV (n=3)
- The ionization energy is the energy required to remove an electron from the ground state (n=1) to infinity (E=0)
- Ionization energy = 0 - E_1 = 0 - (-6.50) = 6.50 eV

Let me formalize this in Lean 4:

```lean4
import Mathlib

/-- The ionization energy of element X, given its first three energy levels:
  E1 = -6.50 eV (n=1), E2 = -3.00 eV (n=2), E3 = -2.00 eV (n=3),
  where the ionization energy is the energy required to remove an electron
  from the ground state to infinity (E = 0 eV). -/
theorem ionization_energy_element_X :
  let E1 : ℝ := -6.50  -- energy of n=1 state in eV
  let E2 : ℝ := -3.00  -- energy of n=2 state in eV
  let E3 : ℝ := -2.00  -- energy of n=3 state in eV
  let E_infinity : ℝ := 0.00  -- energy at infinity (ionized state) in eV
  let ionization_energy : ℝ := E_infinity - E1
  ionization_energy = 6.50 := by sorry
```

Figure 17: Full assistant trace for the atomic energy-level example (panel 2 of 3).

PhyX · full assistant trace | panel 3/3

Same figure, continued in source order. See the prompt figure for input.

3 / 3 · Continue top to bottom

 # Round 1 (Semantic Self-validation)

Your brief analysis:
1. Math Assertion Analysis: The problem asks for the ionization energy of element X. The provided data specifies three energy levels: E1 (n=1) = -6.50 eV, E2 (n=2) = -3.00 eV, E3 (n=3) = -2.00 eV, and the ionized state (infinity) at 0.00 eV. The ionization energy is defined as the energy required to move an electron from the ground state (n=1) to infinity. Mathematically, this is $0.00 - (-6.50) = 6.50$ eV.
2. Lean Statement Analysis (Proof Ignored): The Lean code defines constants E1, E2, E3, and E_infinity with the exact values from the problem (-6.50, -3.00, -2.00, 0.00). It defines `ionization_energy` as `E_infinity - E1`. It asserts the theorem that `ionization_energy` equals 6.50.
3. Comparative Verification: The values in Lean match the problem statement exactly. The definition of ionization energy (`E_infinity - E1`) matches the physical definition for the ground state. The calculated result (6.50) matches the expected mathematical result. The use of `let` bindings in the theorem type is valid Lean 4 syntax for local definitions. The types (ℝ) are appropriate for energy values.
4. Conclusion: The Lean statement is a mathematically equivalent formalization of the natural language problem.
5. Accuracy Confirmation: All variables, values, definitions, and the final assertion correspond exactly between the natural language description and the Lean code.

Result: Correct

Therefore, the outcome of the consistency verification is: Correct

</round>
</think>

<answer>

```lean4
import Mathlib

/-- The ionization energy of element X, given its first three energy levels:
  E1 = -6.50 eV (n=1), E2 = -3.00 eV (n=2), E3 = -2.00 eV (n=3),
  where the ionization energy is the energy required to remove an electron
  from the ground state to infinity (E = 0 eV). -/
theorem ionization_energy_element_X :
  let E1 : ℝ := -6.50  -- energy of n=1 state in eV
  let E2 : ℝ := -3.00  -- energy of n=2 state in eV
  let E3 : ℝ := -2.00  -- energy of n=3 state in eV
  let E_infinity : ℝ := 0.00  -- energy at infinity (ionized state) in eV
  let ionization_energy : ℝ := E_infinity - E1
  ionization_energy = 6.50 := by sorry

```

</answer>

Figure 18: Full assistant trace for the atomic energy-level example (panel 3 of 3).

G.5 PHYSICS: STANDING SOUND WAVE (PHYX)

PhyX · full assistant trace | panel 1/3

Same figure, continued in source order. See the prompt figure for input.

1 / 3 · Continue top to bottom

 # Round 1 (Autoformalization)

<round>

<tool_call>
{"name": "physsearch", "arguments": {"query": "standing wave speed frequency wavelength relationship", "num_results": 8}}
</tool_call>
<tool_response>
[1] ClassicalMechanics.WaveEquation (definition) [physlib]
{d} (f : Time → Space d → EuclideanSpace ℝ (Fin d)) (t : Time) (x : Space d) (c : ℝ) : Prop
The general form of the wave equation describes how a function \(f \) propagates through space and time, where \(c \) represents the propagation speed.
[2] ClassicalMechanics.transverseHarmonicPlaneWave_eq_planeWave (theorem) [physlib]
{c : ℝ} {k : WaveVector} {f₀x f₀y ω δx δy : ℝ} (hc_ge_zero : 0 < c) (hω_ge_zero : 0 < ω)
(hk : k = EuclideanSpace.single 2 (ω / c)) :
(transverseHarmonicPlaneWave k f₀x f₀y ω δx δy hk) =
planeWave
(fun p =>
(f₀x * Real.cos (-(ω / c) * p + δx)) • (EuclideanSpace.single 0 1) +
(f₀y * Real.cos (-(ω / c) * p + δy)) • (EuclideanSpace.single 1 1))
c (WaveVector.toDirection k (by rw [hk]; simp [ne_of_gt, hc_ge_zero, hω_ge_zero]))
The representation of a transverse harmonic plane wave is equivalent to the general plane wave expression when the magnitude of the wave vector \(\|k\| \) is equal to the angular frequency \(\omega \) divided by the propagation speed \(c \).
[3] ClassicalMechanics.planeWave_waveEquation (theorem) [physlib]
(c : ℝ) (s : Direction d) (f₀ : ℝ → EuclideanSpace ℝ (Fin d)) (hf₀ : ContDiff ℝ 2 f₀) :
∀ t x, WaveEquation (planeWave f₀ c s) t x c
A vector-valued plane wave satisfies the wave equation, which describes how a function \(f \) propagates through space and time with a propagation speed \(c \).
[4] ClassicalMechanics.harmonicWave (definition) [physlib]
(a g : ℝ → Space d → ℝ) (ω : WaveVector d → ℝ) (k : WaveVector d) : Time → Space d → ℝ
The function describes a time-harmonic wave in terms of angular frequency \(\omega \) and wave vector \(k \). Specifically, it takes a time \(t \) and a position in space \(r \) to return the wave's amplitude, which is given by the product of a function evaluated at \(\omega k \) and the cosine of the expression \(\omega k \cdot t - g(\omega k) r \).
[5] ClassicalMechanics.planeWave_apply_space_deriv_space_deriv (theorem) [physlib]
{d f₀ c} {s : Direction d} (h' : ContDiff ℝ 2 f₀) (i j : Fin d) :
∂[i] (∂[i] (fun x => planeWave f₀ c s t x j)) = (s.unit i) ^ 2 • fun x => planeWave (iteratedDeriv 2 f₀ ·) c s t x j
For a vector-valued plane wave traveling in the direction of \(s \) with a propagation speed \(c \), the wave equation describes the behavior of the wave in a given space.
[6] HasConstantSpeedOnWith (definition) [mathlib]
{E : Type u_2} → [PseudoEMetricSpace E] → (ℝ → E) → Set ℝ → NNReal → Prop
A function $ f : \mathbb{R} \to E $ has constant speed $ l $ on a set $ s \subseteq \mathbb{R} $ if, for all $ x, y \in s $, the total variation of $ f $ on $ s \cap [x, y] $ equals $ l \cdot (y - x) $.
[7] HasUnitSpeedOn (definition) [mathlib]
(f : ℝ → E) (s : Set ℝ)
A function $ f : \mathbb{R} \to E $ has unit speed on a set $ s \subseteq \mathbb{R} $ if, for all $ x, y \in s $, the total variation of $ f $ on $ s \cap [x, y] $ equals $ y - x $.

Figure 19: Full assistant trace for the standing sound-wave example (panel 1 of 3).

PhyX · full assistant trace | panel 2/3

Same figure, continued in source order. See the prompt figure for input.

2 / 3 · Continue top to bottom

[8] hasConstantSpeedOnWith_iff_variationOnFromTo_eq (theorem) [mathlib]
:
HasConstantSpeedOnWith f s l ↔
LocallyBoundedVariationOn f s ∧ ∀ □x□ (_ : x ∈ s) □y□ (_ : y ∈ s), variationOnFromTo f s x y = l * (y - x)
A function $ f $ has constant speed $ l $ on a set $ s $ if and only if $ f $ has locally bounded variation on $ s $ and for all $ x, y \in s $, the total variation of $ f $ from $ x $ to $ y $ on $ s $ is equal to $ l \cdot (y - x) $.
</tool_response>

<tool_call>
{"name": "web_search", "arguments": {"query": "standing wave displacement nodes distance between consecutive nodes wavelength", "num_results": 5}}
</tool_call>
<tool_response>
[1] Standing waves on strings (video) | Waves
Use the distance between two consecutive nodes or two consecutive antinodes to calculate the length of a standing wave.
https://www.khanacademy.org/science/physics/mechanical-waves-and-sound/standing-waves/v/standing-waves-on-strings
[2] Why is distance between two successive nodes equal to - 1
I'm reading my textbook and it says the distance between two successive nodes is equal to 12λ in standing wave. If λ here means the wavelength ...
https://physics.stackexchange.com/questions/79688/why-is-distance-between-two-successive-nodes-equal-to-frac12-lambda-in-st
[3] Standing wave
The distance between two consecutive nodes or anti-nodes is half the wavelength, λ/2. Halfway between the nodes there are longitudinal displacement anti-nodes ...
https://en.wikipedia.org/wiki/Standing_wave
[4] Standing Waves | 14.4 General Physics
In this lesson Chad provides a lesson on Standing Waves. An introduction first explains the condition of resonance that leads to standing ...
https://www.youtube.com/watch?v=VozE0v5_iMw
[5] Standing wave A standing wave forms when two waves of the ...
The distance between two consecutive nodes (or antinodes) is half the wavelength. Formed due to superposition of two waves traveling in ...
https://www.facebook.com/FDMB47/posts/standing-wavea-standing-wave-forms-when-two-waves-of-the-same-frequency-and-ampl/4260024504325314/
</tool_response>

Based on the physics problem:
- Frequency f = 4.46 × 10^3 Hz
- Separation between ridges (displacement nodes) d = 9.20 cm = 0.0920 m
- The distance between consecutive displacement nodes in a standing wave is λ/2 (half the wavelength)
- Therefore: λ/2 = d, so λ = 2d
- Speed of sound v = f × λ = f × 2d

Let me formalize this in Lean 4:

Figure 20: Full assistant trace for the standing sound-wave example (panel 2 of 3).

PhyX · full assistant trace | panel 3/3

Same figure, continued in source order. See the prompt figure for input.

3 / 3 · Continue top to bottom

```lean4
import Mathlib

/-- Given a standing sound wave in a gas tube where the frequency is 4.46 × 10³ Hz
and the distance between consecutive displacement nodes (ridges of cork filings) is 9.20 cm,
the speed of sound in the gas is f × 2d where d is the node separation. -/
theorem speed_of_sound_from_standing_wave (f : ℝ) (d : ℝ) (v : ℝ)
  (hf : f = 4.46e3)
  (hd : d = 9.20e-2)
  (h_node_separation : d = 1/2 * (v / f)) :
  v = f * 2 * d := by sorry
```

 # Round 1 (Semantic Self-validation)

Your brief analysis:
1. Math Assertion Analysis: The problem asks for the speed of sound v given frequency $f = 4.46 \times 10^3$ Hz and distance between ridges (displacement nodes) $d = 9.20$ cm. The physics principle is that the distance between consecutive nodes in a standing wave is $\lambda/2$, and $v = f\lambda$. Thus $v = 2fd$.
2. Lean Statement Analysis (Proof Ignored): The theorem defines variables f, d, v as real numbers. It assumes $f = 4.46 \times 10^3$, $d = 9.20 \times 10^{-2}$, and the physical relationship $d = \lambda/2$ expressed as $d = 1/2 * (v / f)$. The conclusion is $v = f * 2 * d$.
3. Comparative Verification:
   - Values match (4.46×10^3 and 9.20 cm converted to meters).
   - Physics relationship matches ($d = \lambda/2 \iff d = v/(2f)$).
   - Conclusion matches the derived formula ($v = 2fd$).
   - The Lean statement correctly formalizes the "find X" problem by asserting the equality that defines X.
   - No missing elements or semantic drift.
4. Conclusion: The Lean statement is a mathematically equivalent formalization of the natural language problem.
5. Accuracy Confirmation: All variables, values, units, and physical relationships correspond exactly. The conclusion correctly states the value of the speed of sound derived from the given premises.

Result: Correct

Therefore, the outcome of the consistency verification is: Correct

</round>
</think>

<answer>

```lean4
import Mathlib

/-- Given a standing sound wave in a gas tube where the frequency is 4.46 × 10³ Hz
and the distance between consecutive displacement nodes (ridges of cork filings) is 9.20 cm,
the speed of sound in the gas is f × 2d where d is the node separation. -/
theorem speed_of_sound_from_standing_wave (f : ℝ) (d : ℝ) (v : ℝ)
  (hf : f = 4.46e3)
  (hd : d = 9.20e-2)
  (h_node_separation : d = 1/2 * (v / f)) :
  v = f * 2 * d := by sorry

```

</answer>

Figure 21: Full assistant trace for the standing sound-wave example (panel 3 of 3).