FormalScience: Scalable Human-in-the-Loop Autoformalisation of Science with Agentic Code Generation in Lean

Jordan Meadows[1] **Lan Zhang**[2] **André Freitas**[2,3,4]
[1]Independent Researcher
[2]University of Manchester, UK
[3]Idiap Research Institute, Switzerland
[4]National Biomarker Centre, CRUK-MI, UK
Correspondence: j.c.meadows@hotmail.com

Abstract

Formalising informal mathematical reasoning into formally verifiable code is a significant challenge for large language models. In scientific fields such as physics, domain-specific machinery (*e.g.* Dirac notation, vector calculus) imposes additional formalisation challenges that modern LLMs and agentic approaches have yet to tackle. To aid autoformalisation in scientific domains, we present FormalScience; a domain-agnostic human-in-the-loop agentic pipeline that enables a single domain expert (without deep formal language experience) to produce *syntactically correct* and *semantically aligned* formal proofs of informal reasoning for low economic cost. Applying FormalScience to physics, we construct FormalPhysics, a dataset of 200 university-level (LaTeX) physics problems and solutions (primarily quantum mechanics and electromagnetism), along with their Lean4 formal representations. Compared to existing formal math benchmarks, FormalPhysics achieves perfect formal validity and exhibits greater statement complexity. We evaluate open-source models and proprietary systems on a statement autoformalisation task on our dataset via zero-shot prompting, self-refinement with error feedback, and a novel multi-stage agentic approach, and explore autoformalisation limitations in modern LLM-based approaches. We provide the first systematic characterisation of semantic drift in physics autoformalisation in terms of concepts such as notational collapse and abstraction elevation which reveals what formal language verifies when full semantic preservation is unattainable. We release the codebase together with an interactive UI-based FormalScience system which facilitates autoformalisation and theorem proving in scientific domains beyond physics.[1]

1 Introduction

The informal mathematical reasoning produced by human researchers deviates significantly from the reasoning expressed in formal languages (FL) compiled by automated theorem provers and formal systems (Iancu and Rabe, 2011; McCarthy, 2022; Aliyev, 2024). Formal systems provide strict validation of mathematical statements and proofs at the cost of requiring strict adherence to the syntax of the type-theoretic FL and mathematical library of the prover. This particularly creates barriers towards their broader formalisation in the quantitative sciences (Kaliszyk et al., 2015; Meadows and Freitas, 2021; Bobbin et al., 2023).

The emergence of large language models (LLMs) has partially provided the substrate within which the gap between informal and formal reasoning may be traversed (*i.e.* autoformalisation (Wu et al., 2022)). However, LLMs have shown increasing hallucination rates with greater task complexity (Opedal et al., 2024; Li et al., 2025) or under out-of-distribution shifts (Stolfo et al., 2022; Meadows et al., 2025), which leads a semantic and syntactic gap when transferring informal representations to formal representations (Zhang et al., 2024; Ganguly et al., 2025). As the problem space deviates further from the mathematics most compatible with the FL, the harder it is to build robust LLM-based autoformalisation methods (Zhang et al., 2025c). These critical limitations inhibit the automation of scientific verification, exploration and fact-checking (Lu et al., 2024a; Yamada et al., 2025), making autoformalisation especially challenging in natural sciences such as *Physics*, where semantic drift is significantly magnified by solver incompatibilities with domain notation and calculus.

To understand the bottlenecks of autoformalisation within scientific domains, and using physics as a case study, we:

(i) Introduce `FormalScience` (Figure 1): a lightweight, efficient, and cost-effective human-in-the-loop pipeline for scientific statement-proof generation and semi-autoformalisation, which can convert informal reasoning into largely aligned Lean4

[1]https://github.com/jmeadows17/formal-science

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 23059–23077

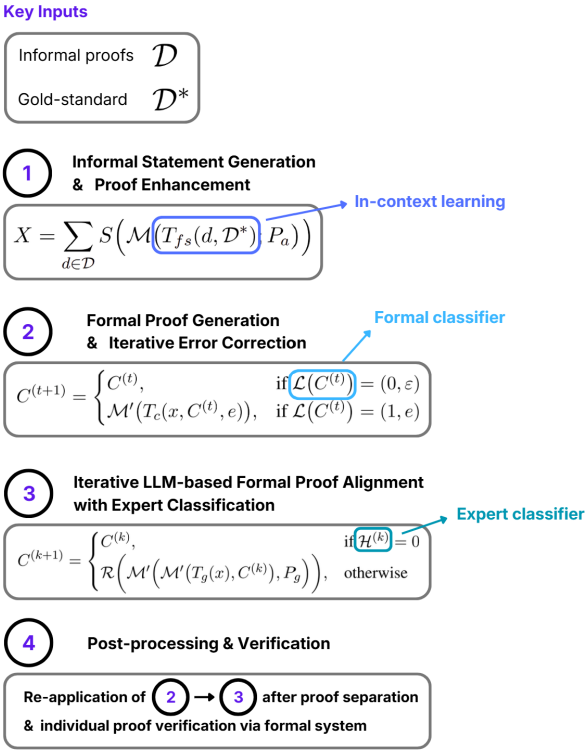

Figure 1: An overview of the `FormalScience` approach.

code with a 100% formal guarantee. The pipeline is domain-agnostic and scalable, enabling the generation of high-quality formal datasets in scientific domains for fine-tuning or evaluating AI systems. In this work, we evaluate FormalScience exclusively on physics.

(ii) Present `FormalPhysics`: a corpus generated from the FormalScience pipeline comprising of 200 university-level physics statements, informal LaTeX proofs, and formal (Lean4) code. Evaluation using LLM judges confirms that FormalPhysics *matches or exceeds* canonical datasets in formal validity, formal quality, and complexity.

(iii) Evaluate open-source (up to 30B parameters) and proprietary models on performing autoformalisation of statements in FormalPhysics with three increasingly complex LLM-based approaches: zero-shot autoformalisation, self-refinement with error feedback, and a novel neuro-symbolic multi-stage agentic approach. The leading methodology (GPT-OSS-20B agent) achieves a formal validity of 31% and competitive alignment scores in statement autoformalisation. All other approaches are subject to a distinct alignment-validity trade-off.

(iv) Provide a detailed qualitative and quantitative analysis of alignment drift in FormalPhysics by characterising drift categories, and using each characterisation to quantitatively describe what Lean4 actually verifies (per physics subdomain) when alignment drift exists yet the formal code successfully compiles, and what formalisation guarantees arise as a consequence. This serves as a step towards explainable verification in scientific autoformalisation.

2 FormalScience: A human-in-the-loop agentic semi-autoformalisation pipeline

Autoformalisation refers to the automatic conversion of an informal content (*e.g.* LaTeX) into formally verifiable code, that can be compiled by a solver or formal system (*e.g.* Lean). However, producing syntactically correct and semantically aligned formalisations without human involvement remains a significant challenge (as we later demonstrate). We propose FormalScience (Alg. 1), a domain-agnostic semi-automatic pipeline for generating problem statements, expanded informal solutions, and formally valid code, thereby facilitating the study of pure autoformalisation. Although applicable to any scientific domain with informal mathematical reasoning (*e.g.* biology, chemistry), all empirical results in this paper are restricted to physics.

We assume a collection of informal proofs $\mathcal{D}$ (*e.g.* LaTeX derivations) and a gold-standard set of informal statement and proofs, $\mathcal{D}^* = [(\mathcal{S}_1, \mathcal{P}_1), ..., (\mathcal{S}_{N'}, \mathcal{P}_{N'})]$ (*e.g.* $N' = 5$). Using in-context learning, a *few-shot template* (T_{fs}) is formulated (see Appendix B) for the purpose of **(1)** generating statements which correspond to the informal proofs in $\mathcal{D}$; and **(2)** altering the proofs in $\mathcal{D}$ to align with the gold-standard $\mathcal{D}^*$. This may for instance include the addition of fine-grained derivation steps or textual context. We can write the resulting intermediate dataset as array concatenation

$$X = \sum_{d \in \mathcal{D}} S\left(\mathcal{M}\left(T_{fs}(d, \mathcal{D}^*); P_a\right)\right) \quad (1)$$

where d is a batch of informal proofs, $\mathcal{M}$ is a multi-turn LLM prompting session, and S is a post-processing function splitting the LLM output $\mathcal{M}\left(T_{fs}(d, \mathcal{D}^*)\right)$ into separate informal statement-proof pairs, such that $x = [(S_1, P_1), ..., (S_{B'}, P_{B'})]$ (where B' is the batch size, see Alg 1), and $x \in X$ aligns with gold-standard $\mathcal{D}^*$. The fixed prompt P_a is used to assess alignment in the prompting session, where the expert suggests improvements and verifies alignment before the next stage.

Next, to obtain the formal code C for each statement-proof pair, we rely on Lean 4 and its Mathlib toolchain to iteratively attempt compilation and return any fatal error messages. We define the tool $\mathcal{L}$ (*i.e.* Lean compiler) such that:

$$\mathcal{L}(C) = \begin{cases} (0, \varepsilon), & \text{if } C \text{ compiles} \\ (1, \mathrm{e}), & \text{otherwise} \end{cases} \tag{2}$$

where e is the error message for a given iteration, and ε is the empty string. To begin iterative error correction, we initially use a code generation template T_g to prompt an LLM-based agent $\mathcal{M}'$ to use tool $\mathcal{L}$, with correction template T_c. If we define the first output as $C^{(0)} = \mathcal{M}'\big(T_g(x)\big)$, we obtain compilable code via

$$C^{(t+1)} = \begin{cases} C^{(t)}, & \text{if } \mathcal{L}\big(C^{(t)}\big) = (0, \varepsilon) \\ \mathcal{M}'\big(T_c(x, C^{(t)}, e)\big), & \text{if } \mathcal{L}\big(C^{(t)}\big) = (1, e) \end{cases} \tag{3}$$

which terminates when $t^* = \min\{t : \mathcal{L}\big(C^{(t)}\big) = (0, \varepsilon)\}$, to give $C = C^{(t^*)}$. In practice, the multi-turn conversational agent $\mathcal{M}'$ operates over a representation of the chat history contained within its context window. We can write this correction loop in shorthand as $C = \mathcal{R}\big(C^{(0)}\big)$.

The previous phase does not consider whether semantic drift occurs between the informal and formal representations. Defining the LLM-based evaluation of the formal code's alignment with the initial prompt $(T_g(x))$ as $\mathcal{M}'\big(T_g(x), C\big)$, we can write the *expert alignment classification* step at iteration k as

$$\mathcal{H}^{(k)} = \mathcal{H}\Big(\mathcal{M}'\big(T_g(x), C^{(k)}\big)\Big) \in \{0, 1\} \tag{4}$$

where the human intervenes as a binary classifier (analogous to compilation tool $\mathcal{L}(C)$). We obtain both *aligned* and *corrected* code via the following iterative process

$$C^{(k+1)} = \begin{cases} C^{(k)}, & \text{if } \mathcal{H}^{(k)} = 0 \\ \mathcal{R}\Big(\mathcal{M}'\big(\mathcal{M}'\big(T_g(x), C^{(k)}\big), P_g\big)\Big), & \text{otherwise} \end{cases} \tag{5}$$

which terminates when $k^* = \min\{k : \mathcal{H}^{(k)} = 0\} \in [0, \mathcal{P}]$, where patience $\mathcal{P}$ denotes the maximum number of expert alignment classification iterations, the fixed prompts P_a and P_g are respectively used provide an LLM-based assessment of alignment and generate improvements, and the code is iteratively corrected via $\mathcal{R}$ (Eq. 3).

Finally, a post-processing step is used to extract individual proofs from each output generated by Eq. 5, which results in tuples (S, P, C). However, the post-processing may have introduced errors, so all C are recompiled with $\mathcal{L}$ to find invalid formal proofs, which are iteratively improved via Eq. 3-5.

3 FormalPhysics: Physics Formalization in Lean4

We select physics as the target scientific domain and Lean as the target formal language, and apply FormalScience using GPT-5.1 and Claude-Opus-4.5 (Anthropic, 2025a). The resulting dataset (FormalPhysics) is a corpus containing 200 physics statements, informal LaTeX proofs, and complete formal proofs. This scale is consistent with established autoformalisation test sets such as miniF2F (Zheng et al., 2022) (244 test examples) and ProofNet (Azerbayev et al., 2023a) (371 examples), while containing approximately twice as many mathematical objects and formulae per example (Table 2). FormalPhysics is intended as an evaluation benchmark rather than a fine-tuning corpus. The examples used as input to FormalScience to generate FormalPhysics are sourced from related work (Meadows et al., 2024), and the human-in-the-loop pipeline was conducted by one physics expert within one month at a total cost of approximately 50 USD. With a motivated group of experts using FormalScience, they could generate thousands of verified *field-specific* formalisations from scientific works in a similar timeframe. Such larger-scale data could be used to fine-tune custom LLMs and improve or evaluate the capabilities of state-of-the-art mathematical discovery approaches. The exact implementation of Alg. 1 used to construct FormalPhysics is described in Appendix A.

3.1 Benchmark Comparison

We compare the properties of our dataset with existing benchmarks for formal mathematics. All datasets in Table 1 contain a formal language statement (s_{FL}) (*e.g.* Lean4 statement), which is the formal representation of a given natural language statement (s_{NL}). *Statement Autoformalisation* is the task of automatically translating the NL statement to the FL statement ($s_{\mathrm{NL}} \to s_{\mathrm{FL}}$), yet only 7/8 datasets contain an NL statement. A natural language proof (p_{NL}) is an informal proof of a given NL statement. Automating this reasoning ($s_{\mathrm{NL}} \to p_{\mathrm{NL}}$) is typical of *scientific/mathematical QA tasks*, yet only 5/8 datasets contain an informal NL proof (*e.g.* LaTeX derivation). Only 3/8

Dataset	Size	Domain	s_{NL}	p_{NL}	s_{FL}	p_{FL}
miniF2F (Zheng et al., 2022)	488	Olympiad (Ol) Math	✓	✓	✓	Partial
ProofNet (Azerbayev et al., 2023a)	371	Undergraduate (UG) Math	✓	✓	✓	×
Lean-Dojo (Yang et al., 2023a)	98,734	Mathlib	×	×	✓	✓
Lean Workbook (Ying et al., 2024)	57,231	High-School Math	✓	×	✓	Partial
FormalMATH (Yu et al., 2025)	5,560	Ol & UG Math	✓	✓	✓	×
Herald-Statement (Gao et al., 2025)	579,883	Mathlib	✓	×	✓	×
Herald-Proof (Gao et al., 2025)	44,553	Mathlib	✓	✓	✓	✓
FormalPhysics	200	Advanced Physics	✓	✓	✓	✓

Table 1: Properties of Lean4 formal benchmarks. s_{NL}: Natural Language Statement; p_{NL}: Natural Language Proof; s_{FL}: Formal Language Statement; p_{FL}: Formal Language Proof.

	s_{NL} Complexity		s_{FL} Correctness		s_{NL}-s_{FL} Alignment	
Dataset	**Objects**	**Formulae**	**FV** (%)	**FQ** (%)	**LP** (%)	**MC** (%)
miniF2F	3.14±1.55	3.21±1.53	88.00	63.00	92.00	92.00
ProofNet	3.67±1.48	3.62±1.52	95.50	61.50	77.50	77.50
Lean Workbook	3.67±1.99	3.62±2.26	89.00	46.00	78.00	85.00
FormalMATH	4.47±2.45	4.53±2.62	97.50	**80.00**	**98.00**	**96.50**
Herald-Statement	4.92±2.43	4.80±2.30	80.50	63.50	87.00	87.00
Herald-Proof	**6.57**±2.32	**6.42**±2.37	2.00	73.00	94.50	94.00
FormalPhysics	6.41±2.34	6.22±2.13	**100.00**	73.50	72.00	72.50

Table 2: Statistics derived from 200 examples randomly selected from each dataset. **Objects**: How many math or physics objects excluding explicit numbers and variables are mentioned directly in the natural language statement? **Formulae**: How many math or physics formulae are mentioned directly in the natural language statement? **FV**: Formal Validity (*i.e.* Pass rate); **FQ**: Formal Quality; **LP**: Logical Preservation; **MC**: Mathematical Consistency.

datasets contain full FL proofs (p_{FL}) (*e.g.* without "sorries"), thereby supporting *automated theorem proving* and *full theorem autoformalisation* tasks. Only FormalPhysics and Herald-Proof are compatible with every task.

We further randomly sample 200 examples from each benchmark and evaluate them across different dimensions, and use elements from a taxonomy for autoformalisation evaluation (Zhang et al., 2025b) to evaluate each sample. Through this dual approach, both the *syntactic quality* of the generated code and its *semantic alignment* with the input natural language (NL) statement are evaluated.

Evaluation methodology. For a given input example, each element can be measured via a *binary classification* with respect to a target characteristic. Classification results are averaged over the sample to give the percentages in Table 2. **Formal Validity** (FV) is judged by the Lean4 theorem prover, while the remaining metrics are assessed via LLM-as-a-judge. **Formal Quality** (FQ) is the rate that the formal code is of high quality in regard to structural clarity and usefulness. **Logical Preservation** (LP) is the rate that the code captures the logical structure and content of the original NL statement. **Mathematical Consistency** (MC) is the rate that the formal code accurately represents mathematical objects and operations present in the NL statement. We also measure the number of objects and formulae in the natural lanugage statements with LLM judges. We prompt GPT-4.1-mini with 0.2 temperature to obtain judgments. Although there might be some noise in the judgments, the underlying LLM is still unbiased towards a specific benchmark and results are still indicative. To assess the robustness of these judgments, we conduct an inter-judge agreement analysis using an independent second judge (Qwen2.5-Coder-7B-Instruct) across $\sim$6,000 paired binary judgments (Appendix D.1). Both judges unanimously rank GPT-5.1 first on every metric in every setting, and the alignment-validity trade-off holds under both judges ($\rho \in [-0.10, 0.30]$, all $p > 0.6$). The 7B judge is systematically more conservative in absolute scores but detects the same underlying quality signal (phi coefficients 0.28–0.37, all $p < 10^{-19}$).

Formalisation in physics is more complex. A given natural language statement in FormalPhysics contains on average *twice as many mathematical objects and formulae* as miniF2F (Zheng et al., 2022), ProofNet (Azerbayev et al., 2023b), and Lean Workbook (Ying et al., 2024). FormalPhysics

also contains $\approx 33\%$ more objects/formulae than the recent FormalMath (Yu et al., 2025) and Herald-Statement (Gao et al., 2025) datasets per example, matched only by Herald-Proof (Table 2).

High formal validity with examples generated through FormalScience. All formal code examples generated from our pipeline are syntactically valid on the latest Lean4 version (see FV, Table 2). Notably, miniF2F and ProofNet were released circa 2023 yet respectively score 88% and 96% successful compilation rates (on an older version of Lean). Herald-Proof (Gao et al., 2025) achieves only 2% FV, while FormalMATH (Yu et al., 2025) scores 98%. FormalPhysics obtains a perfect score.

High formal quality (FQ), low logical preservation (LP) and mathematical consistency (MC). FormalPhysics obtains the second highest score for formal quality (a reference-free evaluation of formal code quality), while scoring the lowest for both logical preservation and mathematical consistency with the NL statement. This is a natural consequence stemming from the fact that around half of NL statements within FormalPhysics contain vector calculus (*i.e.* electromagnetism) or Dirac notation (*i.e.* quantum mechanics) with non-commutative operators. Lean4 does not directly support vector calculus or Dirac notation, and struggles with basic derivatives and integrals (Bobbin et al., 2023), hence alternative strategies and notation were required to successfully compile the code. Formalising physics is extremely challenging (Kaliszyk et al., 2015) and we qualitatively explore such misalignment in Section 5.

4 Experimental Results

Towards fully automated and formally verifiable scientific reasoning, we use FormalPhysics for statement autoformalisation with both open-source and proprietary LLMs in three increasingly complex inference pipelines: **(1)** zero-shot prompting; **(2)** self-refinement with error feedback; **(3)** an agentic code generation approach based on a recent framework (Wang et al., 2024). The results are provided in Table 3.

Models. We use various open-source models such as Qwen2.5-Coder-7B (Hui et al., 2024), DeepSeek-Prover-V2-7B (Ren et al., 2025), Kimina-Autoformalizer-7B (Wang et al., 2025a), GPT-OSS-20B (OpenAI et al., 2025), a distillation of Claude-Sonnet-4.5 (Anthropic, 2025b) onto Qwen3-14B (TeichAI / Liontix, 2025; Yang et al., 2025a), Qwen3-Coder-30B, in addition to the proprietary GPT-5.1 (OpenAI, 2025).

LLM	FV (%)	FQ (%)	LP (%)	MC (%)
(1) `Zero-Shot Autoformalisation`				
Qwen2.5-Coder-7B	1.00	15.00	24.00	20.50
DeepSeek-Prover-7B	13.00	23.00	27.50	24.00
Kimina-7B	**51.50**	6.50	10.50	9.50
GPT-OSS-20B	4.50	68.50	73.00	72.50
GPT-5.1	14.50	**79.50**	**76.50**	**77.00**
(2) `Self-Refinement with Error Feedback`				
Qwen2.5-Coder-7B	1.00	16.50	23.00	19.50
DeepSeek-Prover-7B	4.50	17.00	23.00	23.00
Kimina-7B	**23.00**	6.50	9.50	8.00
GPT-OSS-20B	7.50	70.50	77.00	79.00
GPT-5.1	17.00	**82.50**	**82.00**	**82.00**
(3) `Agentic Code Generation Pipeline`				
Qwen3-Sonnet-14B	**52.00**	1.00	10.50	6.50
GPT-OSS-20B	31.00	**73.00**	**72.50**	**73.00**
Qwen3-Coder-30B	5.50	49.50	59.00	48.00
(4) `FormalScience` (ours)				
GPT-5.1 / Claude-4.5	**100.00**	73.50	72.00	72.50

Table 3: Performance of LLM-based approaches on the FormalPhysics corpus (using GPT-4.1).

LLM	FV (%)	FQ (%)	LP (%)	MC (%)
(1) `Zero-Shot Autoformalisation`				
Qwen2.5-Coder-7B	1.00	8.00	9.00	12.50
DeepSeek-Prover-7B	13.00	12.50	13.50	14.00
Kimina-7B	**51.50**	11.00	14.50	6.50
GPT-OSS-20B	4.50	15.50	12.50	17.50
GPT-5.1	14.50	**27.00**	**28.00**	**33.00**
(2) `Self-Refinement with Error Feedback`				
Qwen2.5-Coder-7B	1.00	11.50	7.00	10.50
DeepSeek-Prover-7B	4.50	26.50	11.50	17.00
Kimina-7B	**23.00**	6.00	7.50	5.00
GPT-OSS-20B	7.50	14.50	10.50	16.50
GPT-5.1	17.00	**38.00**	**35.00**	**42.00**

Table 4: Performance of LLM-based approaches on the FormalPhysics corpus (using Qwen2.5-Coder-7B-Instruct).

4.1 Zero-shot and self-refinement pass

We discuss physics autoformalisation baselines for settings **(1)** and **(2)** (prompts in Appendix B). The self-refinement baselines are based on the zero-shot formalisations from the same LLM.

Trade-off between formal validity and semantic alignment. The Spearman and Pearson coefficients of FV and the mean of FQ, LP, and MC are both zero to one decimal place (with $p > 0.9$), indicating that an approach with high probability of generating syntactic valid formalisations for

physics will struggle to simultaneously represent the intended semantics of the problem.

Invariance to naive prompting. The LLM-as-a-judge scores (FQ, LP, MC) are effectively unchanged per model between the zero-shot and self-refinement settings under the primary GPT-4.1-mini judge. The error-based self-refinement method uses error details to improve zero-shot output, at the cost of $\approx$ 2x the token usage, without clear improvement to formal validity or alignment scores for FormalPhysics. However, this invariance is judge-dependent: under an independent 7B judge (Appendix D.1), GPT-5.1 gains +9.0pp and Kimina-7B drops -4.5pp between settings.

4.2 Agentic code generation

To establish a best-effort open-source baseline on consumer-grade hardware, we implement an agentic code generation pipeline system aiming to maximise compilation rates and alignment without human intervention. We provide a full derivation of our implementation and further details in Appendix C and Alg. 2. Each baseline in Table 3 required **100+ hours of compute on a 5090 RTX GPU** (*i.e.* 30+ minutes per Physics proof). This is approximately equivalent to the code generation rate of the FormalScience approach.

We use an LLM as the base model for a CodeAgent within the smolagents framework (Wang et al., 2024). The agent may use Python functions as tools during inference (in a ReAct (Yao et al., 2022) cycle) which generally features a planning step, a tool-calling action step, and an observation step where the model assesses the tool's output. The agent may output a final answer based on the observation or begin another cycle.

Our implementation features two primary stages. First, an initial generation phase outputs Lean code which is fed to a surface guard that rejects code containing forbidden tokens, incomplete proofs, or malformed imports before compilation. Second, an iterative correction phase compiles the code and categorises errors. *Structural* errors (syntax, unknown identifiers, missing modules) trigger full regeneration using the base LLM with hints based on error type, while *semantic* errors (type mismatches, unsolved goals) are addressed using a patch agent that applies minimal unified diffs. It terminates after 25 correction iterations (full ReAct cycles) or successful compilation in Lean.

The 7B models (Kimina-Autoformalizer, DeepSeek-Prover) were excluded from the agentic setting due to insufficient base capability. DeepSeek-Prover-7B actually *decreases* to 4.5% FV under self-refinement, suggesting it cannot effectively incorporate error feedback even in the simplest iterative setting. Kimina-7B achieves high FV (51.5%) but the lowest alignment scores (FQ: 6.5%, LP: 10.5%), indicating it exploits compilation shortcuts without capturing physics semantics. The multi-step planning, error categorisation, and diff generation required by the agentic pipeline would compound rather than resolve these limitations.

Open-source models can overcome the alignment-validity trade-off. The previous GPT-OSS-20B baselines can only produce less than 10% formally valid formalisations. The agentic *approach improved this to* 31% without any significant decrease in LLM-as-a-judge scores.

Autoformalisation is an emergent capability dependent upon parameter count, neuro-symbolic integration, and test-time scaling. FormalScience obtained a formal validity score of over 3x the best open-source agentic approach. Furthermore, the results are highly sensitive to the base LLM choice, where larger models (*e.g.* 14B, 30B) do not necessarily outperform the naive prompting approaches utilising smaller models (*e.g.* Kimina-7B). *Without iterative dialogue with a symbolic prover*, it is difficult for a leading transformer-based LLM to produce physics formalisations with high formal validity, regardless of the test-time scaling techniques used. Similarly, there exists a minimum base LLM reasoning capability (determined by parameter count, context window, etc.) required to make effective use of symbolic tools (e.g. minimal unified diff) in agentic pipelines. Without any/minimal test-time scaling (*e.g.* zero-shot) GPT-5.1 obtained only 15% formal validity, yet when used within FormalScience it performed significantly better. These three extremes (no symbolic tools, low base LLM intelligence, no test-time scaling) demonstrate the type of experimental optimisation problem required to deliver physics autoformalisation pipelines. The qualitative separation between large and small models persists across judges, though the magnitude of the gap is judge-dependent (Appendix D.1).

5 Alignment and Qualitative Analysis

When a syntactically valid formalisation uses fundamentally different mathematical objects, *what exactly has been verified?* What partial guarantees can formalisation provide when full semantic preservation is unattainable? To answer this we quantitatively measure alignment divergence using a distinct categorisation schema, and use this to guide a qualitative characterisation of the semantic drift induced by formalisation to Lean. We define the following drift categories (visualised in Fig. 2):

Notational Collapse: Domain-specific physics notation collapsed to simpler mathematical objects.

The relevant Fig. 2 example expects a solution integrating over the continuous delta function and multiple substitution operations with terms defined using Dirac/braket notation. The Lean proof correctly defines x as a real scalar but inappropriately defines the quantum state *vector* $|\Psi\rangle$ as the complex scalar Ψ. Quantum mechanics is formalised within a complete, complex inner-product (*i.e.* Hilbert) space, which is enforced implicitly through Dirac notation. When $|\Psi\rangle$ is collapsed to Ψ in this manner (*i.e.* ignoring $\langle x|\Psi\rangle = \Psi(x)$) the fundamental formalism of QM is not respected.

What did Lean verify? Essentially $z \cdot z^* = |z|^2$ for $z \in \mathbb{C}$. All quantum mechanical formalism (and calculus) is absent.

Abstraction Elevation: Symbolic operations replaced by abstract algebraic properties.

The Fig. 2 example requires the evaluation of a definite line integral in 3-dimensional space given the assumption that the scalar potential $\phi(r) = 0$ as $r \to \infty$. The Lean proof correctly defines a real function ϕ with real scalars q (charge) and r (radius), but skips all vector calculus.

What did Lean verify? That $x - 0 = x$ for $x \in \mathbb{R}$. The logic is that $U = q\phi(r)$ (goal statement), so the potential difference between a charge at r and at ∞ is $\Delta U = q\phi(r) - q\phi(\infty) = q\phi(r) - 0 = q\phi(r)$. The physics has been abstracted into the hypothesis statement given in the question.

Proof Strategy Substitution: Theorem proved via alternative approach to the informal derivation.

The informal solution (Fig. 2b) applies a direct partial differentiation operator to a complex function (*i.e.* a single operation). Lean uses an alternative (correct) strategy to show that if two functions are equal their derivatives must be equal.

What did Lean verify? The target statement $\frac{\partial^2 v}{\partial x^2} = -\frac{\partial^2 u}{\partial x \partial y}$, but circumvented direct differentiation of the supporting premise.

Implicit Premise Selection: Assumptions that are unstated in the NL statement or derivation, yet are explicitly defined as FL hypotheses.

The question requires the differentiation of a Lagrangian via the chain rule. Lean provides *a deeper proof by surfacing a number of implicit premises* (*e.g.* $1 + y' > 0 \; \forall y \in \mathbb{R}$, do not divide by zero). This reveals the hidden logical structure of the argument often ignored by physicists.

What did Lean verify? Lean effectively verified $\partial_x(\sqrt{1+x^2}) = x(1+x^2)^{-1/2}$ *without using the chain rule* by introducing relevant premises.

What is Lean verifying quantitatively? Fig. 3 describes the prevalence of each semantic drift category by Physics subdomain. Notational Collapse is present in $> 75\%$ of all QM proofs. This is the most severe type of drift because fundamental scientific context is ignored due to the mistranslation of semantically dense mathematical objects. These proofs verify a *subset* of the required argumentation using *mathematical objects with simple types and structure*.

Abstraction Elevation occurs in $\approx 25\%$ across FormalPhysics. Meaningful physics calculations are replaced by simple abstract arguments often proving trivial results (*e.g.* $x - 0 = x$). In addition, Tab. 8 describes a pattern where vector calculus identities are replaced by abstract linear maps. When vector calculus is formalised as abstract linear algebra, then compilation *verifies algebraic coherence within an abstract vector space*. This does not respect specifics such as coordinate geometry or *e.g.* verify Maxwell's equations. These proofs *verify that a solution is possible* but not necessarily the goal solution outlined in the question.

Proof Strategy Substitution is present within $\approx 33\%$ of FormalPhysics. These proofs *verify statements where the original mathematical objects are preserved* while the goal conclusion *is* verified using a different strategy.

Implicit Premise Selection is the only unambiguously beneficial alignment drift and occurs in $\approx 25\%$ of examples, where only 2% are "pure" (no other drift present). The pure examples are actually

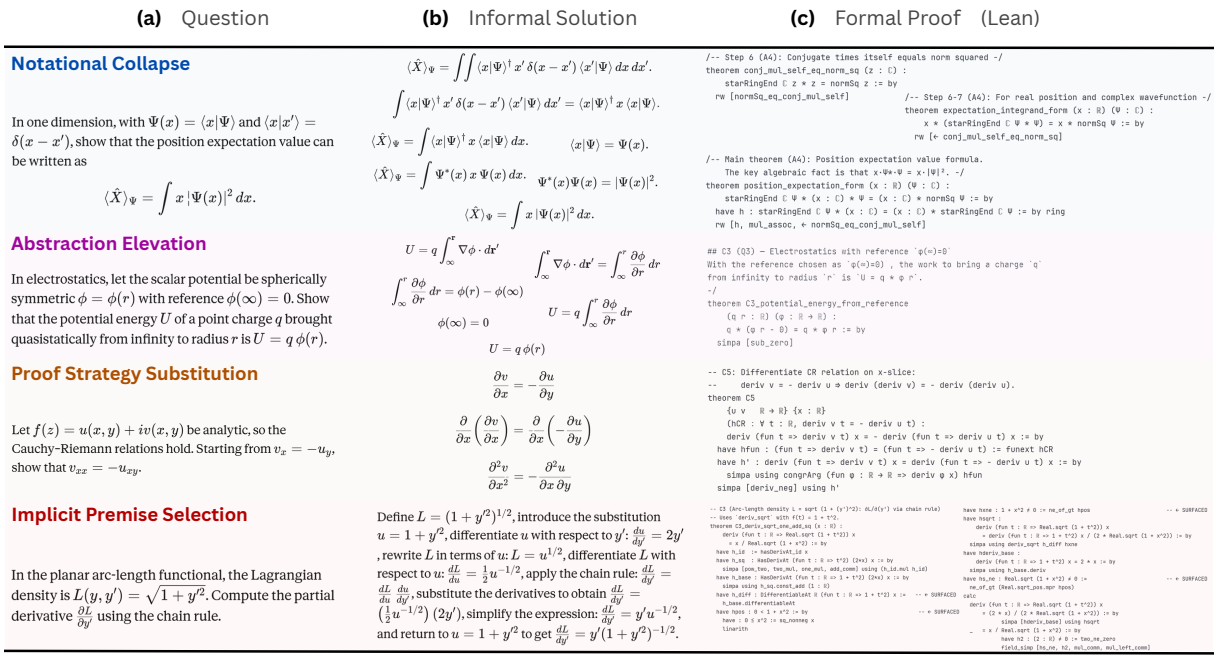

Figure 2: Examples sourced from the FormalPhysics corpus generated via the FormalScience approach applied to Physics. Each row is associated with a different class of semantic drift (*e.g.* Notational Collapse, Abstraction Elevation). See Tab. 8 for a detailed analysis and additional examples.

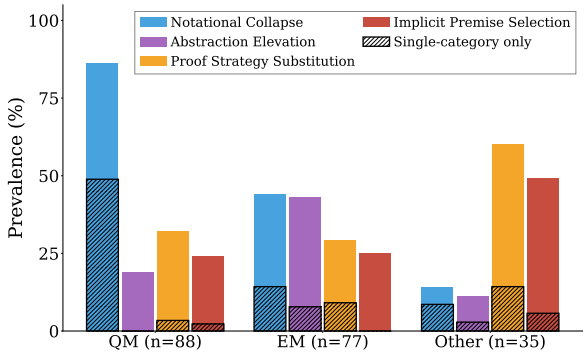

Figure 3: Proportion of alignment drift categories present in FormalPhysics examples by Physics subdomain: Quantum Mechanics (QM), Electromagnetism (EM), and Other (*e.g.* classical, statistical). **Single-category only**: Prevalence of examples featuring only one type of drift.

of *higher quality* than their informal counterparts.

6 Related Work

Recent advances in large language models (LLMs) have renewed interest in autoformalisation, which aims to bridge informal mathematical language and formal proof systems (Wu et al., 2022; Yang et al., 2025b; Mensfelt et al., 2025; Zhang et al., 2025a). Prior work has demonstrated the feasibility of translating natural-language mathematical statements into formal representations and proofs. Autoformalisation has been applied to a range of tasks, including verification of natural-language explanations in natural language inference (Quan et al., 2024b,a) and the construction of automated theorem proving pipelines (Jiang et al., 2023; Tarrach et al., 2024; Liu et al., 2025b). Recent systems increasingly incorporate retrieval-augmented generation, which improves correctness and consistency by leveraging existing formal libraries, in both Isabelle (Zhang et al., 2024) and Lean (Yang et al., 2023b; Liu et al., 2025a; Wang et al., 2025b; Zhang et al., 2025d). Complementary approaches include process-driven frameworks that structure the formalization pipeline (Lu et al., 2024b) and self-consistency methods for selecting high-quality outputs (Li et al., 2024).

On the benchmarking side, existing datasets exhibit a trade-off between quality and scale. Human-curated benchmarks (Zheng et al., 2022; Azerbayev et al., 2023a; Tsoukalas et al., 2024; Poiroux et al., 2025) offer high-quality annotations but are too small for large-scale training. Benchmarks derived from formal libraries (Yang et al., 2023a; Zhang et al., 2024; Xin et al., 2025) scale better but risk data contamination due to overlap with model pretraining. Automated data generation pipelines (Jiang et al., 2024; Ying et al., 2024; Yu

et al., 2025; Gao et al., 2025; Liu et al., 2025c) address scalability but often produce shallow or low-quality formalizations. Our work targets this gap by aiming to improve or characterise the quality of autoformalisation data while preserving scalability and minimising contamination.

We note that existing state-of-the-art autoformalisation systems such as DRIFT (Zhang et al., 2025d) and retrieval-based approaches are evaluated on Mathlib-derived benchmarks where library coverage is comprehensive. FormalPhysics targets domains (vector calculus, Dirac notation, non-commutative operators) where Mathlib support is absent, making direct comparison methodologically inappropriate. Crucially, many related methods do not perform semantic alignment evaluation, which is a central contribution: demonstrating that formal validity alone is insufficient for physics autoformalisation.

7 Conclusion

For the purpose of exploring autoformalisation limitations in science, we propose a human-in-the-loop agentic methodology (`FormalScience`) for formalising scientific reasoning in Lean. Applying it to physics, we produce a dataset (`FormalPhysics`) comprising 200 questions, informal solutions, and formal proofs across quantum mechanics, electromagnetism, and other subdomains. We use compilation success rates and LLM-as-a-judge metrics to compare the formal quality and alignment of formalised statements across several notable autoformalisation benchmarks, and find FormalPhysics leads with 100% formal verification rate (and is competitive in other metrics) at the cost of alignment drift. We argue alignment issues are due to the incompatibility of formal systems with domain-specific machinery such as vector calculus and Dirac notation.

We use FormalPhysics to test the Physics autoformalisation and question-answering capabilities of three increasingly complex LLM-based inference pipelines, including zero-shot, error-driven self-refinement, and full agentic code generation involving notation surface guards, prompt regeneration based on categorised Lean compilation errors, and iterative self-refinement via a patch agent utilising the ReAct (Yao et al., 2022) framework. We test GPT-5.1 and various open-source LLMs (up to 30B parameters) on a consumer-grade GPU to estimate the gap between open-source agentic methodologies and leading human-in-the-loop autoformalisation pipelines. The prevailing open-source agent (using GPT-OSS-20B) obtained a compilation rate $\approx 1/3$ that of the FormalScience approach.

We characterise semantic formalisation errors by defining alignment drift categories, explore errors qualitatively, then determine what verification guarantees can be made depending on the specific drift type supported by domain-specific quantitative analysis. Despite the perfect formal verification rate, notational collapse occurs in most QM proofs, which instead guarantees the verification of surrogate solutions involving simple mathematical objects. Abstraction elevation guarantees meaningful physics computation is circumvented with oversimplified proofs of goal formulae, or is abstracted away with general algebraic proofs which does not necessarily respect the scientific context. We also find drift can be beneficial. In the case of implicit premise selection, without any other drift types, proofs are enriched by supporting premises and rigorous argumentation. Overall, we believe our work will support the development of LLM-based alignment metrics, guide the construction of fully automated formalisation agents, and accelerate scientific formalisation. While the FormalScience pipeline is domain-agnostic, the empirical analysis presented here is restricted to physics; applying FormalScience to other scientific domains (*e.g.* chemistry, biology) remains future work.

Limitations

Dataset scale and scope. FormalPhysics comprises 200 examples focused on quantum mechanics and electromagnetism at university level. This scope, while sufficient for benchmark evaluation, limits generalisability to other physics subdomains (e.g., statistical mechanics, general relativity) and other sciences.

Formal system constraints. As analysed in Section 5, Lean4's Mathlib lacks native support for vector calculus and Dirac notation, necessitating semantic drift in formalisations. The formal proofs therefore verify algebraic consistency within abstract structures rather than the complete physical derivations. Addressing this limitation requires either extending Mathlib's physics coverage or developing physics-specific formal libraries.

Resource requirements. The FormalScience

pipeline required approximately one month of expert effort. The agentic baselines required over 100 hours of GPU compute per 200 examples on consumer hardware. These costs may limit broader adoption and scaling to larger corpora.

Evaluation methodology. Alignment metrics rely on LLM-as-a-judge evaluation, which may not capture all dimensions of semantic preservation. An inter-judge robustness analysis with an independent 7B judge (Appendix D.1) confirms our central findings but reveals that some secondary claims (*e.g.* score invariance under self-refinement, emergence effect size) are judge-dependent. The drift categorisation taxonomy we propose is one principled decomposition but not necessarily complete or unique.

Temporal validity. Results reflect specific LLM versions and Lean4/Mathlib configurations. Model capabilities and library coverage evolve rapidly, and our findings should be interpreted in this context.

References

Yagub N Aliyev. 2024. Set theory and elementary algebra in lean 4 theorem prover. In *2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–6. IEEE.

Anthropic. 2025a. Introducing claude opus 4.5. https://www.anthropic.com/news/claude-opus-4-5. Accessed: 2026-01-01.

Anthropic. 2025b. Introducing claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5. Accessed: 2026-01-01.

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

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023b. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Maxwell P. Bobbin, Samiha Sharlin, Parivash Feyzishendi, An Hong Dang, Catherine M. Wraback, and Tyler R. Josephson. 2023. Formalizing chemical physics using the lean theorem prover. *Preprint*, arXiv:2210.12150.

Debargha Ganguly, Vikash Singh, Sreehari Sankar, Biyao Zhang, Xuecen Zhang, Srinivasan Iyengar, Xiaotian Han, Amit Sharma, Shivkumar Kalyanaraman, and Vipin Chaudhary. 2025. Grammars of formal uncertainty: When to trust llms in automated reasoning tasks. *arXiv preprint arXiv:2505.20047*.

Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. 2025. Herald: A natural language annotated lean 4 dataset. In *The Thirteenth International Conference on Learning Representations*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Mihnea Iancu and Florian Rabe. 2011. Formalising foundations of mathematics. *Mathematical Structures in Computer Science*, 21(4):883–911.

Albert Q. Jiang, Wenda Li, and Mateja Jamnik. 2024. Multi-language diversity benefits autoformalization. In *Advances in Neural Information Processing Systems*, volume 37, pages 83600–83626. Curran Associates, Inc.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. 2023. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*.

Cezary Kaliszyk, Josef Urban, Umair Siddique, Sanaz Khan-Afshar, Cvetan Dunchev, and Sofiene Tahar. 2015. Formalizing physics: automation, presentation and foundation issues. In *International Conference on Intelligent Computer Mathematics*, pages 288–295. Springer.

Sirui Li, Wangyue Lu, Xiaorui Shi, Ke Weng, Haozhe Sun, Minghe Yu, Tiancheng Zhang, Ge Yu, Hengyu Liu, and Lun Du. 2025. Msc-180: A benchmark for automated formal theorem proving from mathematical subject classification. *arXiv preprint arXiv:2512.18256*.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. 2024. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. 2025a. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*.

Qi Liu, Xinhao Zheng, Renqiu Xia, Qinxiang Cao, and Junchi Yan. 2025b. Bootstrapping hierarchical autoregressive formal reasoner with chain-of-proxy-autoformalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xiaoyang Liu, Kangjie Bao, Jiashuo Zhang, Yunqi Liu, Yuntian Liu, Yu Chen, Yang Jiao, and Tao Luo. 2025c. Atlas: Autoformalizing theorems through lifting, augmentation, and synthesis of data. *Preprint*, arXiv:2502.05567.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024a. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, Zhicheng Yang, Jing Tang, and Zhijiang Guo. 2024b. Process-driven autoformalization in lean 4. *Preprint*, arXiv:2406.01940.

John McCarthy. 2022. Artificial intelligence, logic, and formalising common sense. *Machine Learning and the City: Applications in Architecture and Urban Design*, pages 69–90.

Jordan Meadows and André Freitas. 2021. Similarity-based equational inference in physics. *Physical Review Research*, 3(4):L042010.

Jordan Meadows, Tamsin Emily James, and Andre Freitas. 2024. Exploring the limits of fine-grained LLM-based physics inference via premise removal interventions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6487–6502, Miami, Florida, USA. Association for Computational Linguistics.

Jordan Meadows, Marco Valentino, and André Freitas. 2025. Controlling equational reasoning in large language models with prompt interventions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24858–24866.

Agnieszka Mensfelt, David Tena Cucala, Santiago Franco, Angeliki Koutsoukou-Argyraki, Vince Trencsenyi, and Kostas Stathis. 2025. Towards a common framework for autoformalization. *Preprint*, arXiv:2509.09810.

Andreas Opedal, Haruki Shirakami, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. 2024. Mathgap: Out-of-distribution evaluation on problems with arbitrarily complex proofs. *arXiv preprint arXiv:2410.13502*.

OpenAI, :, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, and 108 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

OpenAI. 2025. Gpt-5.1: A smarter, more conversational chatgpt. Large language model.

Auguste Poiroux, Gail Weiss, Viktor Kunčak, and Antoine Bosselut. 2025. Reliable evaluation and benchmarks for statement autoformalization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17958–17980, Suzhou, China. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise Dennis, and Andre Freitas. 2024a. Enhancing ethical explanations of large language models through iterative symbolic refinement. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–22, St. Julian's, Malta. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024b. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2933–2958, Miami, Florida, USA. Association for Computational Linguistics.

Z. Z. Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, Z. F. Wu, Zhibin Gou, Shirong Ma, Hongxuan Tang, Yuxuan Liu, Wenjun Gao, Daya Guo, and Chong Ruan. 2025. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *Preprint*, arXiv:2504.21801.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. 2022. A causal framework to quantify the robustness of mathematical reasoning with language models. *arXiv preprint arXiv:2210.12023*.

Guillem Tarrach, Albert Q. Jiang, Daniel Raggi, Wenda Li, and Mateja Jamnik. 2024. More details, please: Improving autoformalization with more detailed proofs. In *AI for Math Workshop @ ICML 2024*.

TeichAI / Liontix. 2025. TeichAI/Qwen3-14B-Claude-Sonnet-4.5-Reasoning-Distill-GGUF. Model page on Hugging Face. Accessed: 2026-01-01.

George Tsoukalas, Jasper Lee, John Jennings, Jimmy Xin, Michelle Ding, Michael Jennings, Amitayush Thakur, and Swarat Chaudhuri. 2024. Putnambench: Evaluating neural theorem-provers on the putnam mathematical competition. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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, and 21 others. 2025a.

Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *Preprint*, arXiv:2504.11354.

Shaoqi Wang, Lu Yu, Siwei Lou, Feng Yan, Chunjie Yang, Qing Cui, and Jun Zhou. 2025b. Improving autoformalization using direct dependency retrieval. *Preprint*, arXiv:2511.11990.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Norman Rabe, Charles E Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*.

Huajian Xin, Luming Li, Xiaoran Jin, Jacques Fleuriot, and Wenda Li. 2025. Ape-bench i: Towards file-level automated proof engineering of formal math libraries. *Preprint*, arXiv:2504.19110.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin E. Lauter, Swarat Chaudhuri, and Dawn Song. 2025b. Position: Formal mathematical reasoning—a new frontier in AI. In *Forty-second International Conference on Machine Learning Position Paper Track*.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. 2023a. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21573–21612. Curran Associates, Inc.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023b. Leandojo: Theorem proving with retrieval-augmented language models. *Preprint*, arXiv:2306.15626.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. 2024. Lean workbook: A large-scale lean problem set formalized from natural language math problems. In *Advances in Neural Information Processing Systems*, volume 37, pages 105848–105863. Curran Associates, Inc.

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. 2025. Formalmath: Benchmarking formal mathematical reasoning of large language models. *Preprint*, arXiv:2505.02735.

Lan Zhang, Xin Quan, and Andre Freitas. 2024. Consistent autoformalization for constructing mathematical libraries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4020–4033, Miami, Florida, USA. Association for Computational Linguistics.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025a. Autoformalization in the wild: Assessing LLMs on real-world mathematical definitions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1720–1738, Suzhou, China. Association for Computational Linguistics.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025b. Beyond gold standards: Epistemic ensemble of llm judges for formal mathematical reasoning. *Preprint*, arXiv:2506.10903.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025c. MASA: LLM-driven multi-agent systems for autoformalization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 615–624, Suzhou, China. Association for Computational Linguistics.

Meiru Zhang, Philipp Borchert, Milan Gritta, and Gerasimos Lampouras. 2025d. Drift: Decompose, retrieve, illustrate, then formalize theorems. *Preprint*, arXiv:2510.10815.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

A Implementation of FormalScience pipeline to construct FormalPhysics

As described in Alg. 1, the FormalScience pipeline requires in-context examples of informal statements and proofs, and a larger collection of informal proofs we aim to formalise, generate informal statements for, and extend or otherwise alter (*e.g.* add annotations) guided by the few-shot examples.

Data preparation. In the case of the physics examples, we curate 5 such gold-standard statement-proof pairs (see Fig. 4) and randomly select 200 examples from a related dataset of derivations (Meadows et al., 2024). We randomise the 200 examples and group them into batches of 5. Each batch, and the few-shot examples, are now in the required format for input to FormalScience. This ultimately results in 40 few-shot prompts (generated automatically with a Python script) for the initial generation of expanded statements and informal answers (see Section 2) using an LLM.

Stage 1: Generating informal statements and (expanded) proofs. We used GPT-5.1 in thinking mode to generate an intermediate dataset comprising of an informal "question" (statement) and "answer" (proof) from the initial few-shot prompt template (Tab. 5). In particular, the answers were expanded to include NL step annotations in a significant number of cases. After each few-shot prompt, the human expert evaluated the alignment between statement and proof, or otherwise prompted GPT to improve the alignment. Each of the 40 resulting raw LLM outputs contained alternating questions (Q1 - Q5) and answers (A1 - A5). These were split up via a post-processing script to form an intermediate dataset of 40 examples where each example contains 5 dictionaries per derivation. Each dictionary contains the "field" (*e.g.* electromagnetism), the question, and the answer.

Stage 2: Generating formal proofs in Lean4. Each example from the intermediate dataset from the previous stage was input to a formalisation prompt template (see Tab 5).

Our implementation diverges at this point into two complimentary approaches, as we used the ChatGPT interface to produce around 1/3 of examples, and Claude Code (through VSCode) to generate the rest.

The ChatGPT approach required manually copying the outputted formal code, compiling it in Lean, then prompting GPT (within the same session) with the raw compilation errors. We removed non-fatal warnings to reduce context window limitations. Per initial formalisation prompt, this approach required approximately 1-2 hours and several rounds of prompts to generate outputs without compilation errors. Occasionally GPT had to be reminded that incorrect imports did not mean the Lean environment was incomplete.

The Claude Code approach (recommended) included less manual compilation and human intervention. A custom Lean compilation (Python) script was written (*i.e.* $\mathcal{L}(C)$, Section 2), and instructions on how to use it, and handle its output, were appended to the end of the formalisation prompt template. Importantly, due to the context window compactification implemented frequently during this approach, we also saved the exact formalisation prompt in a separate text file in the same folder as the Lean compilation tool, which we found aided the later alignment stage.

Each generated formal code output is prefixed with "C1-C5", due to the formalisation prompt template containing questions "Q1-Q5" and informal solutions "A1-A5".

Stage 3: Iterative alignment. Upon successful compilation without errors, our approach to iterative alignment was centered around a single fixed prompt:

"How well do C1-C5 align with A1-A5, the Requirements, and the Acceptance criteria?"

In either the Claude Code or ChatGPT interface approaches, the human expert evaluates whether the resulting alignment analysis is acceptable. Reaching this point in the pipeline is in itself an iterative process we call *patience*, $\mathcal{P}$, in Alg. 1. We used a maximum patience $\mathcal{P} = 3$ before considering the formal code "well-aligned".

If the formal code was not aligned, an additional fixed prompt (*"Make the suggested improvements and ensure C1-C5 aligns with A1-A5, the Requirements, and the Acceptance criteria."*) was used in either case. Claude Code had to additionally be instructed to iteratively use the Lean compilation tool, while this was manually achieved by the human expert for ChatGPT. This essentially restarts Stage 2 and ticks patience $\mathcal{P}$. Notably, for GPT, we found only compiling the poorly-aligned code vastly accelerated this process.

Stage 4: Post-processing and formal re-verification. Both approaches converge to the same post-processing method. Each formal code output from Stage 3 (a total of 40) contains 5 separate formal code proofs C1-C5, where Mathlib imports for all proofs are combined at the top of the file, within the same block. We use a Python script to separate out all proofs into 5 separate files with identical import blocks. We use another script to unify all questions, informal answers, formal

Q2: Consider a time-independent self-adjoint Hamiltonian $\hat{H}_\lambda$ that smoothly depends on a real parameter λ, with non-degenerate eigenvalue E_λ and eigenvector $|\Psi_\lambda\rangle$, such that Schrodinger holds, *i.e.*

$\hat{H}_\lambda|\Psi_\lambda\rangle = E_\lambda|\Psi_\lambda\rangle$

Prove that

$\frac{\partial E_\lambda}{\partial\lambda} = E_\lambda\langle\frac{\partial\Psi_\lambda}{\partial\lambda}|\Psi_\lambda\rangle + \langle\Psi_\lambda|\frac{\partial\hat{H}_\lambda}{\partial\lambda}|\Psi_\lambda\rangle + E_\lambda\langle\Psi_\lambda|\frac{\partial\Psi_\lambda}{\partial\lambda}\rangle$

A2: If we take the inner product of each side of the Schrodinger equation, we obtain

$\langle\Psi_\lambda|\hat{H}_\lambda|\Psi_\lambda\rangle = E_\lambda\langle\Psi_\lambda|\Psi_\lambda\rangle$

As states are assumed normalized, we recall that $\langle\Psi_\lambda|\Psi_\lambda\rangle = 1$, and hence

$E_\lambda = \langle\Psi_\lambda|\hat{H}_\lambda|\Psi_\lambda\rangle$

If we take the partial derivative with respect to λ we obtain

$\frac{\partial E_\lambda}{\partial\lambda} = \frac{\partial}{\partial\lambda}\langle\Psi_\lambda|\hat{H}_\lambda|\Psi_\lambda\rangle$

Next, via application of the chain rule, we can write

$\frac{\partial E_\lambda}{\partial\lambda} = \langle\frac{\partial\Psi_\lambda}{\partial\lambda}|\hat{H}_\lambda|\Psi_\lambda\rangle + \langle\Psi_\lambda|\frac{\partial\hat{H}_\lambda}{\partial\lambda}|\Psi_\lambda\rangle + \langle\Psi_\lambda|\hat{H}_\lambda|\frac{\partial\Psi_\lambda}{\partial\lambda}\rangle$

Finally, by substituting in the RHS of the Schrodinger equation and factoring out E_λ, we obtain

$\frac{\partial E_\lambda}{\partial\lambda} = E_\lambda\langle\frac{\partial\Psi_\lambda}{\partial\lambda}|\Psi_\lambda\rangle + \langle\Psi_\lambda|\frac{\partial\hat{H}_\lambda}{\partial\lambda}|\Psi_\lambda\rangle + E_\lambda\langle\Psi_\lambda|\frac{\partial\Psi_\lambda}{\partial\lambda}\rangle$

Figure 4: A (human-written) in-context example used in a few-shot prompt to automatically provide physical context to other examples with a similar degree of depth.

proofs, and physics subdomain categories into a dataset of 200 examples.

We compile each formal proof in this dataset to determine if the separation script introduced any errors, making a list of example IDs with new errors. We iteratively improve each proof, beginning from Stage 3 in each case, but reword the alignment prompts to consider only one input question (Q), informal answer (A), and formal code (C) at a time. The resulting examples are added back into the dataset, which finalises our implementation of the FormalScience pipeline used to generate the FormalPhysics dataset.

B Prompts

We provide the few-shot template, the prompt for FormalScience, and prompt for testing LLMs in Figure 4, Table 5, Table 6, respectively.

C Derivation of the agentic code generation pipeline

Let T_g be the initial prompt template for physics question x. The initial generation phase attempts up to $N = 25$ iterations:

$$C^{(0)} = \texttt{extract}\Big(\mathcal{M}\big(T_g(x)\big)\Big) \tag{6}$$

where $\texttt{extract} : \Sigma^* \to \text{Code}$ removes markdown fences from the LLM output. Each candidate $C^{(i)}$ is passed through a surface guard $\mathcal{G} : \text{Code} \to \{0,1\} \times \Sigma^*$ which performs syntax validation before compiling in Lean. The guard returns $(1, \varepsilon)$ if the code passes all heuristic checks, or $(0, r)$ with rejection reason r otherwise. These checks enforce constraints already specified in T_g but which the LLM may violate: absence of forbidden tokens $\mathcal{F} = \{\partial, \nabla, \dot{x}, \dot{y}, \dot{z}, \dagger, \backslash\backslash, \text{`}\}$, absence of incomplete proof markers (`sorry`, `axiom`), balanced comment delimiters, and correct import ordering. If the guard fails, the agent regenerates with feedback:

$$C^{(i+1)} = \texttt{extract}\Big(\mathcal{M}\big(T_r(x, r^{(i)}, \bot)\big)\Big) \tag{7}$$

where $\mathcal{G}(C^{(i)}) = (0, r^{(i)})$, the T_r is a regeneration template that appends the rejection reason to the base prompt, and $\bot$ denotes the absence of a compiler error (since compilation has not yet been attempted). The phase terminates at $i^* = \min\{i : \mathcal{G}(C^{(i)}) = (1, \varepsilon)\}$ or fails if no valid candidate is found within N attempts.

Once initial generation succeeds, the agent enters an iterative compilation loop for up to $N_{\max} = 25$ steps. At each step n, the Lean compiler $\mathcal{L}(C^{(n)})$ returns $(0, \varepsilon)$ on successful compilation or $(1, e)$ with error message e otherwise (*i.e.* Eq. 2). When compilation fails, a categorisation function $\kappa : \Sigma^* \to \mathcal{E}$ maps the error to one of six categories via pattern matching $\mathcal{E} = \{\texttt{syntax}, \texttt{unknown\_id}, \texttt{missing\_module}, \texttt{type\_mismatch}, \texttt{unsolved\_goals}, \texttt{other}\}$. These categories are split into structural errors $\mathcal{E}_{\text{struct}} = \{\texttt{syntax}, \texttt{unknown\_id}, \texttt{missing\_module}\}$ and semantic errors $\mathcal{E}_{\text{sem}} = \{\texttt{type\_mismatch}, \texttt{unsolved\_goals}, \texttt{other}\}$, which determine the repair strategy.

For structural errors, the agent performs full regeneration using the primary LLM:

$$C^{(t+1)} = \texttt{extract}\Big(\mathcal{M}\big(T_r(x, \kappa(e), e)\big)\Big) \tag{8}$$

if $\kappa(e) \in \mathcal{E}_{\text{struct}}$ where T_r now includes both the error category and the compiler message, instructing the model to rewrite the entire file. The regenerated code must again pass the surface guard; if $\mathcal{G}(C^{(t+1)}) = (0, \_)$, the regeneration is discarded and the step is logged as failed.

For semantic errors, the agent uses a specialised patch agent $\mathcal{A}_{\text{patch}}$ equipped with a unified diff tool:

$$C^{(t+1)} = \mathcal{A}_{\text{patch}}\Big(T_p\big(\texttt{number}(C^{(t)}), e\big)\Big) \tag{9}$$

Algorithm 1 FormalScience

Require: Derivation batches $\mathcal{D}$, batch size B (with $B' \in [1, B]$), gold-standard examples $\mathcal{D}^* = [(Q_i, A_i)]_{i=1}^{N'}$, patience $\mathcal{P}$. Here, questions Q and informal answers A are respectively equivalent to informal statements S and informal proofs P in the main text.

Ensure: Formally verified corpus $Z = \{(Q, A, C) : \mathcal{L}(C) = (0, \varepsilon)\}$

Stage 1: Informal QA Generation and Alignment

$X \leftarrow []$
for each batch $d \in \mathcal{D}$ **do**
  $x \leftarrow S\big(\mathcal{M}(T_{fs}(d, \mathcal{D}^*); P_a)\big)$ ▷ Few-shot QA generation with human alignment
  $X \leftarrow X \cup \{x\}$
end for

Stage 2: Code Generation and Iterative Correction

$Y \leftarrow []$
for each $x = [(Q_i, A_i)]_{i=1}^{B'} \in X$ **do**
  $C^{(0)} \leftarrow \mathcal{M}'(T_g(x))$ ▷ Initial code generation
  $t \leftarrow 0$
  while $\mathcal{L}(C^{(t)}) = (1, e)$ **do** ▷ Compilation fails
    $C^{(t+1)} \leftarrow \mathcal{M}'(T_c(x, C^{(t)}, e))$ ▷ Error-guided correction
    $t \leftarrow t + 1$
  end while
  $C \leftarrow C^{(t)}$ ▷ $C = \mathcal{R}(C^{(0)})$

Stage 3: Formal Language Alignment

  $k \leftarrow 0$
  while $\mathcal{H}^{(k)} = 1$ **and** $k < \mathcal{P}$ **do** ▷ Human rejects alignment
    $C' \leftarrow \mathcal{M}'(\mathcal{M}'(T_g(x), C^{(k)}), P_g)$ ▷ Alignment improvement
    $C^{(k+1)} \leftarrow \mathcal{R}(C')$ ▷ Re-verify compilation
    $k \leftarrow k + 1$
  end while
  $y \leftarrow S'(x, C^{(k)})$ ▷ Split into (Q_i, A_i, C_i) tuples
  $Y \leftarrow Y \cup \{y\}$
end for

Stage 4: Post-processing and Final Verification

$Z \leftarrow \text{flatten}(Y)$
for each $(Q, A, C) \in Z$ **do**
  if $\mathcal{L}(C) = (1, e)$ **then** ▷ Post-processing introduced errors
    $(Q, A, C) \leftarrow$ apply Eq. 5 ▷ Re-align and correct
  end if
end for
return Z

if $\kappa(e) \in \mathcal{E}_{\text{sem}}$ where `number` : Code $\to$ Code prepends line numbers and T_p is a patch template requesting a minimal unified diff. The patch agent operates in a ReAct loop, generating a diff and applying it via the `apply_unified_diff` tool. This reflects the intuition that structural errors indicate fundamental misunderstanding requiring complete regeneration, while semantic errors (type mismatches, unsolved goals) are often addressable through localised edits to tactics or expressions.

The correction loop terminates when $\mathcal{L}(C^{(t)}) = (0, \varepsilon)$ or when $n = N_{\max}$. Unlike the human-in-the-loop pipeline (Alg.1, Stage 3), this approach does not perform formal language alignment. The agentic code generation pipeline used to obtain the

Prompt Name	Purpose	Template Structure	Input Variables	Output Expected
Informal Expansion Prompt	Convert equation-only derivations into contextually-rich Q&A pairs	*"The following 5 questions (Q1–Q5) and respective answers (A1–A5) are few-shot examples..."* followed by 5 quantum Q&A pairs, then *"Now, the following* ***equation-only*** *derivations (D6–D10) represent the underlying equational reasoning of a Physics derivation. You must convert each derivation into a* ***physically-correct*** *and* ***contextually-enriched*** *Question (Q6–Q10) and Answer (A6–A10) pair..."* followed by 5 derivations and closing instruction to ensure one equality per equation with correct physical meaning and standard notation.	• 5 few-shot Q&A pairs. • 5 equation-chain derivations (*i.e.* a batch).	5 new Q&A pairs (Q6–Q10, A6–A10) with physics context and properly formatted LaTeX equations
Lean Formalization Prompt	Autoformalize informal physics derivations into compilable Lean 4 + Mathlib proofs	**Header:** Task description requesting compilable Lean 4 proofs without axioms. **Q/A Block:** 5 pairs formatted as *"**Qi:** {question} **Ai:** {answer}"*. **Requirements:** (1) No `axiom`/`sorry`; (2) Use Mathlib theorems; (3) Explicit physics modelling; (4) Single compilable file; (5) Include docstrings and clear theorem names; (6) Prefer `calc` blocks over `simp`; (7) Deterministic rewrites. **Deliverables:** One file with C1–C5. **Acceptance:** Must compile with no axioms.	• 5 expanded informal Q&A pairs from previous LLM output.	Single Lean 4 file containing 5 theorems (C1–C5) with Mathlib imports, docstrings, explicit hypotheses, and complete proofs

Table 5: Prompt Templates Used in the FormalScience Pipeline

relevant baselines is described in Alg. 2.

D Supplementary Analysis

D.1 LLM-Judge Robustness Analysis

To assess the robustness of our LLM-as-a-judge alignment evaluation, we conduct an inter-annotator agreement analysis using a second, independent judge (Qwen2.5-Coder-7B-Instruct) alongside our primary judge (GPT-4.1-mini). Both judges evaluated all 200 items $\times$ 5 models $\times$ 2 settings (zero-shot and self-refinement) $\times$ 3 alignment metrics (FQ, LP, MC), yielding $\sim$6,000 paired binary judgments. Full 7B judge results are reported in Table 4.

Judges detect the same underlying quality signal despite different calibration. Pooling all (item $\times$ model) pairs per setting and metric gives 1,000 paired binary observations per condition. The phi coefficient is positive and highly significant across all six conditions ($\phi \in [0.28, 0.37]$, all $p < 10^{-19}$), confirming both judges respond to the same underlying quality signal.

Disagreement is structured and asymmetric. The 7B judge is systematically more conservative. For the two strongest models (GPT-5.1 and GPT-OSS-20B), over 95% of inter-judge disagreements take the form GPT=True / 7B=False. When the conservative 7B judge accepts an item, the GPT judge almost always agrees (91–100% for GPT-5.1, 89–95% for GPT-OSS-20B). The 7B-positive items therefore form a high-confidence consensus subset.

Model-level rankings are partially preserved. Kendall's τ across all six (metric $\times$ setting) comparisons ranges from 0.2 to 1.0 (median 0.80). Five of six comparisons yield $\tau \geq 0.6$. Both judges unanimously rank GPT-5.1 first on every metric

Task	Content
Zero-Shot Statement Autoformal-isation	You are an expert in formal language Lean4. You will be given a physics statement and its proof written in natural language and LaTeX symbols. Your task is to provide the formal code of the given natural language physics statement and its proof in Lean4 with the following instructions: 1. You should give the formal code directly without any additional comments or explanations. If the given physics statement is a theorem or lemma, omit the formal proof and use the default 'sorry' mode in the formal code. 2. In case that you need to import any necessary preambles, you should not import any fake (non-exist) preambles. 3. You should wrap the formal code in a way illustrated as the following: %%%%%%%%%% Your Formal Code %%%%%%%%%% Strictly follow the instructions that have been claimed. Natural language statement: {nl_statement} Give me the Lean4 formal code of the statement:
Self-Refinement with Error Feedback	You are an expert in formal language Lean4. You will be given a physics statement and its proof written in natural language and LaTeX symbols. You will also be given a formal code which attempted to describe the given physics statement in Lean4. Your task is to refine the given formal code to make it correct while maintaining the alignment with the given natural language physics statement. Here are some instructions for your task: 1. You should give the formal code directly without any additional comments or explanations. If the given physics statement is a theorem or lemma, omit the formal proof and use the default 'sorry' mode in the formal code. 2. In case that you need to import any necessary preambles, you should not import any fake (non-exist) preambles. 3. You should wrap the formal code in a way illustrated as the following: %%%%%%%%%% Your Formal Code %%%%%%%%%% Strictly follow the instructions that have been claimed. Natural language statement: {nl_statement} There are some Lean4 formal codes describing the given physics statement: {formal} You should refine the formal code for your task to make it correct. Here are some feedbacks about the formal code which can be used to help your task: {According to the theorem prover, the error details of the provided formal code are: error_details }

Table 6: Prompts.

Setting	**FQ**	**LP**	**MC**
Zero-Shot	0.28	0.33	0.37
Self-Refinement	0.30	0.32	0.35

Table 7: Phi coefficients between GPT-4.1-mini and Qwen2.5-Coder-7B-Instruct judges (n=1,000 per cell, all $p < 10^{-19}$).

in every setting. Rank instability is confined to the middle and bottom of the ranking, where the 7B judge's compressed score distributions (four models within a 5pp band) make fine-grained distinctions unreliable.

Implications for paper claims. The alignment-validity trade-off (Section 2) is *supported*: under the 7B judge, the Spearman correlation between FV and mean alignment remains near zero ($\rho \in [-0.10, 0.30]$, all $p > 0.6$). The claim that scores are "effectively unchanged with self-refinement" is *judge-dependent*: it holds for the GPT judge but not universally (GPT-5.1 gains +9.0pp under the 7B judge). The emergence effect size is *directionally preserved but magnitude-reduced*: the ratio of GPT-5.1 to DeepSeek-Prover-7B mean alignment is $3.1\times$ under the GPT judge but $2.2\times$ under the 7B judge (zero-shot). Kimina-7B's compilation shortcut exploitation is *supported* under both judges.

Algorithm 2 Agentic Code Generation Pipeline

Require: Physics question x, initial generation template T_g, regeneration template T_r, patch template T_p, max initial attempts $N = 25$, max correction steps $N_{\max} = 25$, forbidden tokens $F = \{\partial, \nabla, \dot{x}, \dot{y}, \dot{z}, \dagger, \backslash\backslash, \text{`}\}$
Ensure: Compilable code C such that $\mathcal{L}(C) = (0, \varepsilon)$, or failure

Stage 1: Initial Generation with Surface Guard

$i \leftarrow 0$
while $i < N$ **do**
  $C^{(i)} \leftarrow \text{extract}\big(\mathcal{M}(T_g(x))\big)$ ▷ Eq. 9: Generate and extract code
  $(g, r^{(i)}) \leftarrow G(C^{(i)})$ ▷ Surface guard validation
  if $g = 1$ **then** ▷ Guard passed
    goto Stage 2
  else
    $C^{(i+1)} \leftarrow \text{extract}\big(\mathcal{M}(T_r(x, r^{(i)}, \bot))\big)$ ▷ Eq. 10: Regenerate with feedback
  end if
  $i \leftarrow i + 1$
end while
return FAILURE ▷ No valid candidate in N attempts

Stage 2: Iterative Compilation and Error Correction

$C \leftarrow C^{(i^*)}$ where $i^* = \min\{i : G(C^{(i)}) = (1, \varepsilon)\}$
$t \leftarrow 0$
while $t < N_{\max}$ **do**
  $(s, e) \leftarrow \mathcal{L}(C^{(t)})$ ▷ Eq. 5: Lean compilation
  if $s = 0$ **then** ▷ Compilation succeeded
    return $C^{(t)}$
  end if
  $\kappa(e) \leftarrow$ CATEGORIZE(e) ▷ Error categorisation

  if $\kappa(e) \in E_{\text{struct}}$ **then** ▷ Structural errors: syntax, unknown_id, missing_module
    $C' \leftarrow \text{extract}\big(\mathcal{M}(T_r(x, \kappa(e), e))\big)$ ▷ Eq. 11: Full regeneration
    if $G(C') = (1, \varepsilon)$ **then** ▷ Guard passes
      $C^{(t+1)} \leftarrow C'$
    else
      Log regeneration failure; $C^{(t+1)} \leftarrow C^{(t)}$
    end if
  else ▷ Semantic errors: type_mismatch, unsolved_goals, other
    $C^{(t+1)} \leftarrow \mathcal{A}_{\text{patch}}\big(T_p(\text{numberlines}(C^{(t)}), e)\big)$ ▷ Eq. 12: Patch agent
  end if
  $t \leftarrow t + 1$
end while
return $C^{(N_{\max})}$ ▷ Return best effort after max steps

– Helper Definitions –

function CATEGORIZE(e)
  $E_{\text{struct}} \leftarrow \{\texttt{syntax}, \texttt{unknown\_id}, \texttt{missing\_module}\}$
  $E_{\text{sem}} \leftarrow \{\texttt{type\_mismatch}, \texttt{unsolved\_goals}, \texttt{other}\}$
  return category $\kappa(e) \in E_{\text{struct}} \cup E_{\text{sem}}$ via pattern matching
end function

function $G(C)$ ▷ Surface guard: G : Code $\to \{0, 1\} \times \Sigma^*$
  if $\exists f \in F : f \in C$ **then return** $(0,$ "forbidden token f")
  end if
  if $\texttt{sorry} \in C \vee \texttt{axiom} \in C$ **then return** $(0,$ "incomplete proof")
  end if
  if $|C|_{\text{/-}} \neq |C|_{\text{-/}}$ **then return** $(0,$ "unmatched delimiters")
  end if
  if imports not correctly ordered **then return** $(0,$ "import ordering")
  end if
  return $(1, \varepsilon)$
end function

Table 8: Detailed semantic drift analysis: Physics notation versus Lean4 formalisation. This table provides extended examples showing how mathematical objects transform during autoformalisation.

ID	Physics (Informal)	Lean4 (Formal)
Pattern A: Quantum Operators → Scalars/Algebra Elements		

4	**Statement:** $N = a^\dagger a = \frac{1}{2}(q - ip)(q + ip)$
	Key objects:
	• q, p: Position/momentum operators with $[q, p] = i$
	• $a, a^\dagger$: Ladder operators (non-commuting)
	• N: Number operator
	Physics content: Operator algebra on Hilbert space
		`theorem C5_number_operator_expand`
		`(q p : C) : -- Complex numbers!`
		`let s := (1 : R) / Real.sqrt 2`
		`let a := s * (q + I * p)`
		`let adag := s * (q - I * p)`
		`adag * a = s*s * (p^2 - I*(p*q)`
		`+ I*(q*p) + q^2)`
		Drift: $[q, p] = i$ *not enforced*; proof holds for any complex numbers

14	**Statement:** Heisenberg uncertainty $\Delta X \cdot \Delta P \geq \hbar/2$
	Key objects:
	• $\|f\rangle, \|g\rangle$: Fluctuation kets in Hilbert space
	• $[\hat{X}, \hat{P}] = i\hbar$: Canonical commutation relation
	• Cauchy-Schwarz on inner products
	Physics content: Fundamental quantum bound
		`theorem C1`
		`{E : Type*} [InnerProductSpace C E]`
		`{f g : E} {hbar : R}`
		`(hbar_nonneg : 0 <= hbar)`
		`(hcomm : inner f g - inner g f`
		`= I * (hbar : C)) :`
		`||f|| * ||g|| >= hbar / 2`
		Drift: Commutator relation is a *hypothesis*, not derived from $[\hat{X}, \hat{P}]$

12	**Statement:** $\frac{d\hat{x}(t)}{dt} = \frac{i}{\hbar}[\hat{H}, \hat{x}(t)]$
	Key objects:
	• $e^{i\hat{H}t/\hbar}$: Unitary time evolution operator
	• $\hat{x}(t) = U(t)\hat{x}U^\dagger(t)$: Heisenberg picture
	• Product rule on operator exponentials
	Physics content: Quantum dynamics
		`theorem C3`
		`{A : Type*} [NormedAlgebra R A]`
		`(c : R) {H x0 : A}`
		`(U W : R -> A)`
		`(hU : forall t, deriv U t = c*(H*U t))`
		`(hW : forall t, deriv W t = -c*(W t*H))`
		`(hUd : forall t, DifferentiableAt R U t)`
		`(hWd : forall t, DifferentiableAt R W t) :`
		`forall t, deriv (s => U s * x0 * W s) t`
		`= c * (H*(U t*x0*W t) - (U t*x0*W t)*H)`
		Drift: Exponential structure assumed via `hU`, `hW`; works in any normed algebra

Pattern B: Vector Calculus → Abstract Linear Maps		

5	**Statement:** $\hat{\mathbf{r}} \cdot \nabla f = \frac{\partial f}{\partial r}$
	Key objects:
	• $\nabla f = \frac{\partial f}{\partial r}\hat{\mathbf{r}} + \frac{1}{r}\frac{\partial f}{\partial \theta}\hat{\boldsymbol{\theta}} + \frac{1}{r\sin\theta}\frac{\partial f}{\partial \varphi}\hat{\boldsymbol{\varphi}}$
	• Spherical coordinate basis $\{\hat{\mathbf{r}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\varphi}}\}$
	Physics content: Coordinate geometry
		`theorem C1_radial_projection`
		`{E : Type*} [InnerProductSpace R E]`
		`{er e_th e_ph : E} {ar ath aph : R}`
		`(hnorm : ||er|| = 1)`
		`(horth_r_th : inner er e_th = 0)`
		`(horth_r_ph : inner er e_ph = 0) :`
		`inner er (ar*er + ath*e_th + aph*e_ph) = ar`
		Drift: Gradient formula becomes hypothesis; spherical coords → abstract orthonormal basis

24	**Statement:** Gauss's law from $\nabla^2\phi = 4\pi G\rho$
	Key objects:
	• $\int_V \nabla^2\phi \, dV = 4\pi G \int_V \rho \, dV$
	• Divergence theorem: $\int_V \nabla \cdot \mathbf{F} \, dV = \oint_{\partial V} \mathbf{F} \cdot d\mathbf{S}$
	Physics content: Field theory
		`theorem C5_gauss_from_poisson`
		`(G : R) (S V1 V2 : R)`
		`(hPoisson : V1 = (4 * pi * G) * V2)`
		`(hDivThm : S = V1) :`
		`S = (4 * pi * G) * V2 := by`
		`simpa [hPoisson] using hDivThm`
		Drift: All integration/divergence content in hypotheses; proof is trivial substitution

Pattern C: Integration → Limit/Algebraic Statement		

10	**Statement:** $W = \int_{r_0}^{\infty} \frac{GMm}{r^2} \, dr = \frac{GMm}{r_0}$
	Key objects:
	• Definite integral of inverse-square force
	• Improper integral at infinity
	Physics content: Work-energy theorem
		`theorem C1`
		`(G M m r0 : R) (hr0 : 0 < r0) :`
		`Tendsto (fun R => (G*M*m)*(r0^-1 - R^-1))`
		`atTop (nhds ((G*M*m) / r0)) := by`
		`have h_inv : Tendsto (R^-1) atTop (nhds 0)`
		`:= tendsto_inv_atTop_zero`
		`...`
		Drift: Integral → limit statement; uses `tendsto_inv_atTop_zero` lemma

27	**Statement:** $U = q\phi(r)$ from $W = \int_{\infty}^{r} \mathbf{F} \cdot d\mathbf{l}$
	Key objects:
	• Line integral of electric force
	• Reference $\phi(\infty) = 0$
	Physics content: Potential energy definition
		`theorem C3_potential_energy_from_reference`
		`(q r : R) (phi : R -> R) :`
		`q * (phi r - 0) = q * phi r := by`
		`simpa [sub_zero]`
		Drift: Entire integration content abstracted; proof is `sub_zero`