QuantumLean-Bench: A Unified Benchmark for Informal and Formal Quantum Reasoning

Isha Goswami [*1] **Anushka Paulchoudhury** [*1] **Robert Joseph George** [1] **Anima Anandkumar** [1]

Abstract

Existing Large Language Model benchmarks typically evaluate either informal reasoning or formal proof generation, and are often limited to just one subject area or type of task. To address this gap in robust benchmarks, we introduce QuantumLean-Bench, where each problem pairs an informal natural-language response with a formal counterpart in Lean, enabling direct evaluation of whether models preserve mathematical structure across reasoning modalities. The benchmark unifies 931 problems spanning chemistry, computing, information theory, physics, and quantum machine learning, drawn primarily from well-known online courses and established instructional resources. We evaluate several frontier and open-source models, including GPT-5.4 and Gemini 3.1, using a task-aware 0/1/2 rubric and report both coverage and full-benchmark performance. We find that while frontier models achieve near-ceiling performance on informal reasoning (GPT-5.4: 0.981 normalized average, 96.2% strict accuracy), performance drops substantially in the formal setting (0.621 normalized average, 36.1% strict accuracy). These results highlight a persistent gap in current model capabilities: strong natural-language reasoning does not reliably translate into precise formal representations.

1. Introduction

Benchmarking Large Language Model (LLM) reasoning has become an active area of research as model capabilities have rapidly improved (Wei et al., 2022; Kaplan et al., 2020). LLMs have emerged as powerful tools not only for language tasks, but also for reasoning tasks that require domain-specific knowledge and have achieved expert-level performance (Waqas et al., 2025). LLM reasoning abilities are particularly interesting to study in quantum science contexts, where problems demand multi-step reasoning over narrow yet mathematically complex domains(Wei et al., 2023). Work exploring the use of LLMs in scientific discovery (Zhang et al., 2025a) motivates improvement of their capabilities in specific scientific domains.

Despite these rapid advancements, the ”black box” nature of LLMs necessitates rigorous evaluation to distinguish reasoning from sophisticated pattern matching. Benchmarks serve as the primary compass for the research community to identify specific areas where model performance collapses despite high surface-level accuracy, thus tracking progress of and guiding model development over time (Bohacek et al., 2025). Emergent behaviors, such as chain-of-thought reasoning, often appear unpredictably as models scale, making standardized testing valuable for ensuring reliability (Bowman, 2023). Without such frameworks, it remains difficult to verify if a model has internalized the underlying logical structures or is merely overfitting to its training data.

Moreover, formal reasoning, specifically in the functional programming language, Lean, requires structured syntax rules that are readily subject to formal verification (lea, 2026b). As a result, a standard, unified benchmark containing advanced tasks would serve as a critical foundation for systematically comparing LLM capabilities (Liang et al., 2023a). Quantum scientific fields are particularly suited to this task because they are mathematically well-structured and central to modern science. Since quantum fields also rely heavily on informal reasoning, they have not been fully formalized in proof assistants. There exists a persistent translation challenge between the way results are derived in practice and how they must be expressed for formal verification (Douglas et al., 2026). This setting provides a natural testbed for examining the gap between informal scientific reasoning and formal verification in the context of LLMs.

Lean cleanly separates syntax from semantics and provides a medium through which informal and formal scientific reasoning can be directly compared within a single, unified framework. Since proofs are symbolic, train and test splits can be created on premises, and proof reuse can be

[1]California Institute of Technology, Pasadena, CA, USA. Correspondence to: Isha Goswami <igoswami@caltech.edu>, Anushka Paulchoudhury <anushka@caltech.edu>.

Proceedings of the AI4Physics Workshop at the 43rd International Conference on Machine Learning (AI4Physics@ICML 2026), Seoul, South Korea. 2026. Copyright 2026 by the author(s).

detected. Moreover, datasets can be split at the level of assumptions or lemmas because proofs in formal language are more like data objects rather than prose. Work such as LeanDojo (Yang et al., 2023), LeanAgent (Kumarappan et al., 2025), Lean Copilot (Song et al., 2025), Lean Progress, and Lean IDE has already shown how Lean is integrated into a growing ecosystem of LLM evaluation tools (lea, 2026a). While Lean provides a strong guarantee of correctness, the formalization process that requires each theorem to be broken down into minute steps remains a time-consuming task for humans. This is a space where LLMs can excel and will enable a rigorous feedback loop where we can assess where LLMs succeed, fail, and generalize beyond memorized proof patterns.

For each quantum topic, we assembled a set of problems in the following categories: numerical problems, proof prompts, and algorithmic questions; across the following topics: quantum computing, quantum physics, quantum information theory, quantum cryptography, and quantum machine learning. This project aims to facilitate an interdisciplinary understanding of AI reasoning evaluation by providing a standard set of tasks that bridge text-based reasoning, symbolic mathematics, and executable computing to test both informal and formal reasoning capabilities of LLMs.

2. Related Work

2.1. LLM Reasoning and Automated Theorem Proving

Several approaches have been proposed to enhance LLM performance on theorem-proving tasks, evolving from basic generative models to sophisticated, closed-loop verification systems. GPT-f was one of the earliest examples of a neural theorem-proving approach applied to proofs in the Metamath formal system (Polu & Sutskever, 2020). DeepTheorem uses reinforcement learning with rewards for better proof reasoning and intermediate steps rather than just a final answer, outperforming standard supervised finetuning (Zhang et al., 2025b). On the other hand, InternLM2.5-StepProver targets formal proof search on MiniF2F using a learned critic to rank proof trajectories, resulting in better performance than baseline theorem-proving systems (Wu et al., 2025; Zheng et al., 2022). Despite these advancements, a disconnect remains between informal intuition and formal rigor for benchmarking LLM reasoning on scientific tasks.

2.2. LLMs and Formal Proof Assistants

Beyond informal reasoning benchmarks, a growing body of work explores the integration of LLMs with formal proof assistants, which provide machine-verifiable guarantees of correctness for generated proofs. This integration is supported by foundational environments as mentioned previously: tools such as LeanDojo, LeanAgent, and Lean Copilot provide environments and APIs that enable LLMs to generate tactics, explore proof states, and interact with Lean's verification engine during automated proof search (lea, 2026a). TheoremLlama is trained on an NL-FL-aligned dataset: formal language proofs that have natural language comments in them (Wang et al., 2024). Moreover, AxProver gives LLMs direct access to Lean tools rather than relying on finetuning, performing better than most frontier LLMs on math reasoning benchmarks like PutnamBench (Breen et al., 2025). These methods are increasingly extended beyond mathematics: PhysProver applies reinforcement learning on DeepSeek-Prover-V2-7B to generate Lean-verifiable physics conjectures and introduces PhysLeanData, a formalized physics dataset spanning classical and modern physics domains (Zhang et al., 2026). Despite these advances, current benchmarks remain concentrated in mathematics and physics, with limited coverage of broader scientific domains and little focus on quantum science.

2.3. Existing Benchmarks

We note that our benchmark is not the first one for quantum science fields in a single benchmark ecosystem. The application of AI to quantum science has traditionally focused on hardware optimization. Benchmarks such as QASMBench (Li et al., 2023), SupermarQ (Tomesh et al., 2022), and QPack (Mesman et al., 2022) evaluate circuits, compilers, devices, or application-level execution metrics rather than natural-language tasks. In parallel, a recent line of work has begun evaluating LLMs on quantum code generation, including Qiskit HumanEval (Vishwakarma et al., 2024), QHackBench (Basit et al., 2025), QuanBench (Guo et al., 2025), and QCoder Benchmark (Mikuriya et al., 2025). Related systems such as the Qiskit Code Assistant train LLMs on Qiskit-based quantum programming tasks (**?**), while Liang et al. propose Quantum GPT-Guided Architecture Search for LLM-driven ansatz design in quantum algorithms (Liang et al., 2023b). The closest prior work on quantum-science reasoning is QuantumBench (Minami et al., 2025), which evaluates LLMs on approximately 800 multiple-choice questions across nine quantum domains, and the more focused quantum mechanics benchmark of Rithvik (Rithvik, 2026), which studies physics problem solving in a single domain. Both of these are important quantum science benchmarks showing interest in LLM evaluation on these topics, but remain informal-only. Finally, formal math theorem-proving benchmarks such as miniF2F (Zheng et al., 2022), and FormalMATH (Yu et al., 2025) demonstrate that Lean-grounded evaluation is indispensable for understanding formal reasoning. Yousefzadeh et al. propose that a complete, rigorous benchmark should include formal problem statements, informal problem statements, formal proofs, and informal proofs

Table 1. How QuantumLean-Bench compares to representative prior benchmark families. The claim is deliberately narrow: it concerns *multi-domain quantum-science reasoning with paired formal outputs*, not all quantum benchmarking in general.

Benchmark family	Representative examples	Multi-domain	Open-ended	Lean/ formal	Paired tasks
Systems and hardware benchmarks	QASMBench, SupermarQ, QPack	✓ Yes	✗ No	✗ No	✗ No
Chemistry and QML datasets	MoleculeNet, QDataSet	Usually no	✗ No	✗ No	✗ No
Quantum code-generation benchmarks	Qiskit HumanEval, QHackBench, QuanBench, QCoder	Mostly no	Limited	✗ No	✗ No
Quantum reasoning benchmarks	QuantumBench, Rithvik (2026)	Partial	✓ Yes	✗ No	✗ No
Formal theorem-proving benchmarks	miniF2F, LeanDojo, FormalMATH	✗ No	Limited	✓ Yes	✗ No
QuantumLean-Bench	**This work**	✓ Yes	✓ Yes	✓ Yes	✓ Yes

(Yousefzadeh & Cao, 2025). The holistic approach remains rare in literature, but one notable exception is the Gaokao-Formal benchmark that has natural-language mathematical proofs paired with human-verified Lean4 proofs from the Chinese entrance exam (Xuejun et al., 2025).

This existing work suggests significant interest and progress in studying LLM reasoning across various science topics. However, these efforts remain fragmented: no single framework jointly evaluates formal and informal reasoning on a common benchmark, and existing studies are typically confined to a single domain—predominantly mathematics or physics. Existing formal systems are also task-specific or require specialized setups. This gap highlights an opportunity for a unified benchmark integrating both formal and informal verifiable reasoning tasks. Our work addresses exactly this by introducing QuantumLean-Bench, a quantum + X benchmark that evaluates LLM reasoning across diverse science domains, using numerical, algorithmic, and proof-based problems.

3. Methods

3.1. Datasets

QuantumLean-Bench is an aggregation of diverse problems from publicly available datasets and instructional materials drawn from established online courses. The package we release contains 11 machine-readable files organized based on quantum science topics and the source.

- Qiskit QuantumKatas (Krishna, 2023): This is a dataset of programming exercises derived from Microsoft's Quantum Katas quantum computing course (Corporation). The exercises in this dataset use the open-source software framework, Qiskit, developed by IBM for building, simulating, and running quantum programs (ibm, 2026). This dataset was available on HuggingFace, so the problems were extracted with a Python script and copied over directly to our repository.
- QM7 (qm7, 2017): The QM7 dataset contains the Coulomb matrices of 7,165 molecules and their corresponding atomization energies. This dataset was used to train machine learning models to simulate quantum chemistry by predicting the atomization energy from the Coulomb matrix, so these questions all fell into the 'numerical' category. This dataset was available for direct download from the website, so a Python script was used to read the `.mat` file.
- MIT OpenCourseware (mit, 2026): MIT OpenCourseWare is a well-known open-access platform that provides high-quality course materials, including problem sets and exams from the Massachusetts Institute of Technology. We use problems from the most recent and highest-level (graduate versus undergraduate) courses, when available. Specifically, the problems in our dataset were selected from the following courses:

  5.61 Physical Chemistry (undergraduate, Fall 2007)

  6.845 Quantum Complexity Theory (graduate, Fall 2010)

  18.435J Quantum Computation (graduate, Fall 2003)

  6.5630 Advanced Topics in Cryptography (graduate, Fall 2023)

  MAS.865J Quantum Information Science (graduate, Spring 2006)

  5.73 Introductory Quantum Mechanics I (graduate, Fall 2005)

  The MIT OpenCourseWare problems were provided as PDF files on the website. We used an LLM to extract and convert them into a LaTeX-formatted representation suitable for direct prompting. The extracted questions were then manually verified against the original PDFs to ensure fidelity.

Additionally, since quantum machine learning is a relatively new area without standardized textbooks or canonical benchmark sources, we generated QML questions by curating representative topics from recent literature, including variational quantum circuits, quantum kernels, data encoding, and trainability. These topics were then converted into prompts structured for benchmarking aligned with the same numerical, algorithmic, and proof-based schema used across the dataset. Quantum cryptography has limited representation in the dataset because it is a relatively new and narrow subfield of quantum science. To prioritize the quality of the questions, we restricted our sources to those that are credible and established. Although broader collections of quantum science problems were available, many lacked sufficient documentation, suggesting poor verifiability in their construction or curation. A breakdown of the question types and topics is shown in Table 2.

Subject	Total	Numerical	Algorithmic	Proof-based
Chemistry	306	204	37	65
Computing	220	12	132	76
Cryptography	4	0	3	1
Information Theory	86	15	26	45
Physics	215	118	54	43
Machine Learning	100	23	31	46
Total	**931**	**372**	**283**	**276**

Table 2. Distribution of problems across subjects and categories.

For reporting, derivation, conceptual, computational, and open-ended entries are mapped into the closest numerical, algorithmic, or proof-based family.

Numerical challenges require a quantitative answer or prediction from a well-specified mathematical or physical setup. These problems typically involve evaluating expressions, matrix elements, expectation values, or probabilities.

For the proof category of questions, the model is to derive, verify, or justify a general statement about a system. These problems emphasize structured reasoning and using formal properties of quantum systems. Topics included are deriving properties of quantum algorithms, formalizing quantum information theoretic inequalities, or verifying algebraic identities involving operators on Hilbert spaces.

Algorithmic tasks ask for the implementation or reasoning of code, circuits, or algorithmic structures based on existing canonical quantum algorithms and protocols. Unlike numerical tasks, the output is typically a structured object such as a set of allowed transitions, a circuit, or pseudocode. The focus for these questions is on procedural correctness and completeness rather than formal justification. A breakdown of problem types for each subject is shown in Figure 1.

The benchmark is intentionally heterogeneous to more effectively evaluate a model's ability to generalize across diverse modes of quantum reasoning, rather than overfitting to a single, narrowly defined format. Chemistry and physics contribute most of the numerical load; computing contributes the largest concentration of algorithmic problems; information theory and QML contribute many proof-oriented tasks.

Each benchmark entry contains a problem statement, associated metadata, an informal response dictionary, a formal response dictionary, and a manual_eval field. Since evaluation data is highly fragmented, with results spread across papers, leaderboards, and different frameworks in incompatible formats, a key design choice for the project was adopting a structured, universal schema. We chose to follow the schema proposed by the "Every Eval Ever" initiative for a unified schema that standardizes how evaluation data is recorded, including inputs, outputs, metadata, and metrics (eve, 2026). Many entries also include a reference solution when it was available from the source. For exact-answer benchmarks such as QM7, this reference serves as a ground-truth target. For open-ended problems, it functions as contextual information for calibration rather than a strict output target. This structure enables both exact-match evaluation and more qualitative, manually assessed judgment, with the full schema shown in Appendix A.

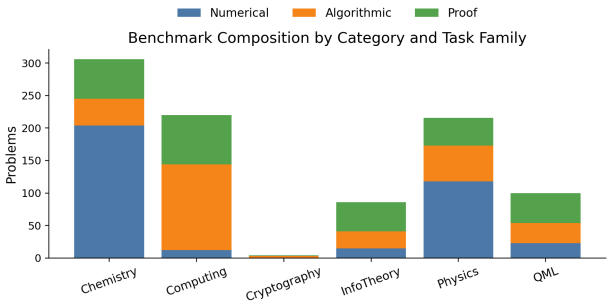

Figure 1. Number of problems of each type across subjects

3.2. Prompting and Evaluation

For each response type, the API prompt specified instructions to produce a clear, direct answer containing only the solution. A representative prompt for informal responses was used, with a closely analogous formulation applied for formal responses as shown in Appendix B.

All reported scores are based on manual review. For each model response, we assign a score in $\{0, 1, 2\}$, where 2 indicates a substantially correct and useful response, 1 indicates a partially correct, incomplete, or under-justified response, and 0 indicates a wrong, missing, vacuous, or otherwise unusable response. To summarize performance across the benchmark, we report both a normalized average score and

a strict accuracy. If n_0, n_1, and n_2 denote the counts of responses receiving scores 0, 1, and 2 across N problems, then

$$\text{NormalizedAvg} = \frac{2n_2 + n_1}{2N}, \qquad \text{StrictAcc} = \frac{n_2}{N}.$$

The normalized average gives partial credit, while strict accuracy counts only fully correct responses. A detailed scoring rubric is shown in Appendix C. Rubric-based evaluation of LLM outputs has been shown to significantly improve human alignment, and therefore correctness, since scores must be justified by criteria like coherence and logic (Liu et al., 2023). By assessing dimensions such as logical soundness and constraint satisfaction, we create a rigorous feedback loop serving as an informal compiler for these quantum science tasks.

Coverage matters because not every model was run on every benchmark file, and some model families fail to produce usable answers altogether. We therefore define coverage as the fraction of benchmark problems with a non-empty response in the corresponding slot. In all full-benchmark aggregates, missing responses are assigned a score of 0. This is especially important for formal reasoning, where a model that does not produce a usable formalization should not appear competitive simply because it attempted fewer tasks.

The scoring rubric is task-aware. For numerical tasks, full credit requires the requested final quantity or exact target value rather than only a descriptive discussion. For algorithmic tasks, full credit requires an explicit construction such as a circuit, protocol, or code-level procedure together with enough justification to establish correctness. For proof-based tasks, full credit requires a coherent chain of reasoning that reaches the requested conclusion rather than gesturing at the correct intuition. The formal slot is graded analogously, but with emphasis on whether the response faithfully encodes the requested claim or construction in Lean code.

We evaluate formal responses primarily with a manual faithfulness rubric rather than requiring Lean compilation for credit: a response can preserve the intended mathematical structure, assumptions, and useful proof skeleton even if it does not compile in isolation. We also report raw Lean compilation as an auxiliary diagnostic under the repository's pinned Lean environment: GPT-5.4 compiles on 407/931 formal responses (43.7%), while StepFun compiles on 2/7 non-empty formal attempts, DeepSeek on 0/404, Qwen-3.5-4B on 3/404, and Gemini on 9/190.

For this release, GPT-5.4 was evaluated on all 931 benchmark problems in both the informal and formal slots. StepFun was also run on nearly the full benchmark for informal responses, but produced only a very small number of non-empty formal responses. DeepSeek, Qwen-3.5-4B, and Gemini 3.1 were evaluated only on a subset of benchmark files, so their results should be interpreted as subset-only baselines rather than direct end-to-end competitors.

In addition to aggregate metrics, we track performance at the level of individual benchmark slices, including category (chemistry, computing, etc.) and task family (numerical, algorithmic, proof-based). This is important because aggregate scores can obscure systematic weaknesses. For example, a model that performs well on descriptive or proof-oriented prompts may still fail consistently on exact-answer numerical tasks, or may produce incomplete constructions on algorithmic problems.

We also explicitly separate quality from attempt. A model that produces a large number of partial or low-quality responses should be distinguished from a model that fails to produce any usable output at all. This distinction is particularly important in the formal slot, where some models either emit empty responses or produce syntactically plausible but semantically invalid code. By reporting both coverage and full-benchmark scores, we ensure that models are evaluated on both their willingness to attempt problems and the correctness of those attempts.

Finally, we emphasize that the paired informal–formal structure is not simply a formatting variation. The two slots represent different reasoning regimes. The informal slot evaluates whether a model can explain or derive a result in natural language, while the formal slot evaluates whether the same reasoning can be made explicit, structured, and in principle machine-checkable. This distinction allows us to directly measure the gap between explanation and formalization, rather than inferring it from separate benchmarks.

4. Results

4.1. Overall Performance

The main quantitative result is that frontier models are already very strong on informal quantum reasoning, but substantially weaker on formalization of the same tasks. GPT-5.4 achieves a normalized average of 0.981 and a strict accuracy of 96.2% on the informal side of the full benchmark. However, on the formal side, the same model drops to a normalized average of 0.621 and a strict accuracy of 36.1%. This informal–formal gap is the central empirical pattern of our benchmark.

StepFun exhibits a different profile. Informally, it answers almost the entire benchmark, with 99.2% coverage, but the quality is much lower: 0.481 normalized average and 15.1% strict accuracy. Formally, it effectively collapses. Only 7 of 931 formal responses are non-empty, and all of them receive a score of 0 under our faithfulness-based rubric. This suggests that weak formal performance is not merely a

formatting issue, but often reflects an inability to produce a meaningful formalization at all.

Importantly, this gap is not explained by a lack of coverage or missing attempts. GPT-5.4 achieves full coverage on both informal and formal slots, indicating that the drop in performance reflects a genuine difference in task difficulty rather than a failure to respond. The formal degradation therefore, isolates the challenge of preserving mathematical structure under stricter representational constraints.

This pattern is also consistent across both exact-answer and open-ended tasks. For the same problem, models that perform near-perfectly in natural-language reasoning frequently fail to encode the same result as a precise formal object. This suggests that current models rely heavily on implicit reasoning shortcuts that do not translate cleanly into explicit formal representations.

The subset-only baselines are consistent with the same overall picture. Qwen-3.5-4B outperforms DeepSeek on both the informal and formal subsets that were evaluated, while Gemini performs especially poorly on the informal subset. None of these models approaches GPT-5.4 on full-benchmark quality, and none resolves the core gap between informal reasoning and formal reasoning.

4.2. Score Distributions

Aggregate averages compress the three-level rubric into a single scalar, so it is useful to inspect the underlying score distribution as well. GPT-5.4 is near ceiling on informal tasks, with no score-0 informal responses across the full benchmark. Most of its residual error on the informal side comes from score-1 cases that are relevant but incomplete. On the formal side, however, the distribution shifts substantially: a large portion of responses receive partial credit rather than full credit, indicating that the model often captures some of the desired structure without fully formalizing the requested result.

StepFun shows a different pattern. Its informal responses are dominated by score-1 answers, meaning that it often identifies the right topic or approach but fails to complete the task to the level required for full credit. Its formal responses are uniformly score 0, reflecting empty, truncated, or otherwise unusable formal outputs.

These distributions highlight that most formal errors are not catastrophic failures but incomplete ones. A large fraction of GPT-5.4's formal responses receive score 1, indicating that the model often identifies the correct structure but fails to fully encode the requested claim or construction. This suggests that the primary bottleneck is not knowledge, but completeness and precision.

By contrast, StepFun's distribution reflects a different failure mode. Its informal responses frequently fall into the score-1 category, indicating partial understanding, while its formal responses are overwhelmingly scored 0. This combination points to a breakdown earlier in the reasoning pipeline, where the model struggles to convert high-level intuition into either complete informal answers or usable formal artifacts.

4.3. Variation Across Categories and Task Families

Performance also varies systematically across both subject areas and problem families. Chemistry and physics contribute the largest share of numerical tasks, while computing contributes the largest share of algorithmic tasks. Information theory and QML contain many proof-oriented prompts. This heterogeneity is useful because it prevents benchmark performance from collapsing into a proxy for a single narrow task type.

On the informal side, GPT-5.4 is consistently strong across all categories, with scores near the ceiling in chemistry, cryptography, and QML. On the formal side, performance remains meaningfully lower across every category. The strongest formal results for GPT-5.4 occur in chemistry and physics, while computing, information theory, and QML remain more difficult to formalize faithfully. For weaker models, performance becomes much more uneven: some categories receive full coverage but poor quality, while others receive little or no usable output at all.

A similar pattern appears when grouping problems into numerical, algorithmic, and proof-based families. GPT-5.4 performs best on informal numerical and proof-oriented tasks, and its formal numerical performance remains stronger than its formal algorithmic or proof-oriented performance. This suggests that exact-target tasks may still be easier to encode formally than open-ended derivations or multi-step constructions, even when the same model performs well informally on all three. The subset-only baselines follow the same broad pattern but at a lower absolute level.

Physics and computing tasks exhibit distinct formalization failure modes due to differences in reasoning structure. Physics problems are largely numerical or derivational and show strong informal performance, but formal responses often remain incomplete: models capture the correct equations or relationships yet omit assumptions or fail to finish derivations, leading to many partial-credit outcomes. In contrast, computing tasks are primarily algorithmic and require explicit constructions, and the dominant failure mode is incomplete implementation: models identify the correct approach but fail to produce valid circuits, protocols, or procedures that satisfy all constraints. Subset-evaluated models follow the same pattern at lower quality, with partially correct derivations in physics and high-level but non-executable descriptions in computing. As a result, physics tasks are

Model	Informal cov.	Informal avg.	Informal strict	Formal cov.	Formal avg.	Formal strict
GPT-5.4	100.0%	0.981	96.2%	100.0%	0.621	36.1%
StepFun	99.2%	0.481	15.1%	0.8%	0.000	0.0%
DeepSeek	100.0%	0.210	2.1%	72.1%	0.180	0.0%
Qwen-3.5-4B	100.0%	0.266	12.6%	72.1%	0.209	5.6%
Gemini	100.0%	0.010	0.0%	29.3%	0.052	0.2%

Table 3. Overall benchmark results. Coverage is computed over all 931 problems. Scores are full-benchmark aggregates, so missing responses contribute zero.

Model/slot	n_0	n_1	n_2	Notes
GPT-5.4 informal	0	35	896	Near-ceiling informal performance with few partial-credit cases
GPT-5.4 formal	111	484	336	Many relevant but incomplete formalizations
StepFun informal	177	613	141	Many responses begin correctly but stop early
StepFun formal	931	0	0	Missing formal responses count as score 0; only 7 were non-empty

Table 4. Score distributions for the two models evaluated end-to-end on the full benchmark.

relatively easier to formalize than computing tasks, largely because numerical problems admit more direct encoding, whereas algorithmic tasks require multi-step structure and explicit correctness conditions.

One consistent trend is that numerical tasks remain relatively easier to formalize than algorithmic or proof-based tasks. Even though all three task families are grounded in the same underlying quantum concepts, numerical tasks often admit a direct encoding as a value or equation, while algorithmic and proof-based tasks require representing multi-step structure, constraints, and intermediate reasoning. This added structural burden appears to amplify the informal–formal gap.

We also observe that category-level differences often disappear once task family is taken into account. For example, chemistry appears difficult in aggregate, but much of this difficulty arises from its concentration of exact-answer numerical tasks. Similarly, computing tasks become more challenging in the formal setting because they require explicit constructions rather than descriptive explanations. This suggests that task format, rather than subject area, is the primary driver of difficulty in the benchmark.

4.4. Qualitative Patterns

The quantitative results become easier to interpret once we inspect the responses that received each score. One recurring pattern is plausible but unfinished reasoning. This appears frequently in lower-performing informal responses, especially on computing and information-theory tasks, where a model may identify the correct gate family, protocol intuition, or physical principle, but fail to produce the final construction or conclusion demanded by the prompt.

A second recurring pattern is the distinction between exact-answer and open-ended tasks. In exact-target datasets such as QM7, scientific fluency without the final numerical answer is still wrong. By contrast, in many proof-oriented physics or information-theory prompts, a response can earn substantial credit by organizing the key ideas correctly even if every algebraic step is not fully explicit. This difference is one reason chemistry is especially punishing for weaker models.

The third recurring pattern is the failure mode of formalization itself. Partial formal credit usually does not arise from random syntax errors. Instead, it often comes from a recognizable but weakened formalization: helper definitions, simplified analogues, or theorem statements that point toward the right idea without actually encoding the requested claim. This is important because a response that preserves some of the correct mathematical structure is much more useful than one that is empty or completely off target, even if neither compiles in its current form.

Another important qualitative distinction is between surface-level correctness and structural correctness. Many responses that receive partial credit are locally correct—they reference the right equations, operators, or physical principles—but fail to organize these elements into a complete solution. This distinction is especially important for formal reasoning, where correctness depends not only on the presence of the right components, but on how they are composed.

We also observe that some failure modes are amplified by the cross-domain nature of the benchmark. Tasks that combine elements from multiple areas (e.g., physics intuition with algorithmic construction) are more likely to produce incomplete or inconsistent responses. This suggests that the ability to integrate multiple reasoning styles remains a

challenge even when each individual component is handled well in isolation.

5. Discussion and Limitations

Our results are most informative when category, task family, coverage, and the informal versus formal split are all visible at once. These findings suggest that performance on quantum-science benchmarks depends strongly on how tasks are represented. Informal reasoning and explanation tasks tend to elicit strong performance, while structured formalization exposes substantial weaknesses. This separation indicates that current models are better characterized as producing plausible scientific narratives than consistently correct formal artifacts. The three task families: algorithmic, numerical, and proof-based, reveal a coarse hierarchy of formalization difficulty. While the task families were initially chosen to ensure heterogeneity, our results suggest that task format may be a more significant driver of formalization difficulty than subject area, with numerical tasks formalized more successfully than algorithmic or proof-based ones.

At the same time, the benchmark has several clear limitations. First, the released manual judgments were produced through a careful but ultimately single-annotator process, so we do not yet report inter-annotator agreement. Second, the category sizes are imbalanced: chemistry and physics contribute far more examples than cryptography, which limits the granularity with which smaller subject areas can be analyzed. Third, the primary formal slot is judged for faithfulness rather than compilation. We include raw Lean compilation as an auxiliary diagnostic, but compilation alone is too brittle to capture whether a response preserves the intended mathematical structure. Finally, the dataset is intentionally heterogeneous: it combines numerical labels, textbook-style derivations, coding exercises, and theorem-like tasks. That breadth is central to the benchmark's purpose, but it also means that no single automatic metric can fully replace careful interpretation.

One way to interpret this gap is that current models are optimized for plausibility rather than fidelity. Informal reasoning allows models to rely on implicit assumptions, approximations, and narrative coherence, all of which can produce convincing explanations without guaranteeing correctness. Formalization, by contrast, forces these assumptions to be made explicit. Any missing step, unstated constraint, or weakened claim becomes immediately visible.

This has implications beyond benchmarking. In scientific and engineering workflows, the ability to produce correct explanations is valuable, but the ability to produce verifiable artifacts is often critical. The gap we measure therefore, reflects a practical limitation: models that appear highly capable in natural-language settings may still require substantial human intervention to produce reliable formal outputs.

Finally, our results suggest that improving formal reasoning will likely require more than scaling existing models. The failure modes we observe—partial formalizations, weakened claims, and missing constraints—indicate that models need stronger mechanisms for tracking structure, dependencies, and completeness. Addressing these issues may require advances in training data, objective functions, or hybrid neuro-symbolic approaches.

6. Conclusion

Our benchmark supports three broader conclusions. First, frontier models have become highly capable informal reasoners on quantum-science tasks. At least for the graduate- and advanced undergraduate-level problems represented in this benchmark, informal reasoning is no longer the main bottleneck for the strongest models. Second, formalization remains substantially harder than explanation. The drop in GPT-5.4 performance from near-ceiling informal reasoning to much lower formal performance is large enough to be a stable and meaningful empirical signal rather than a small implementation artifact. Third, subject labels alone do not explain model performance. Once we distinguish between numerical, algorithmic, and proof-based tasks, many of the apparent category effects become easier to interpret.

Overall, the benchmark is designed to measure a question that is becoming increasingly important in scientific AI: whether a model can move from quantum-science explanation to quantum-science formalization without losing the structure of the task. Across 931 problems and six subject areas, there is not a straightforward answer. Informal reasoning is already strong for the best models we evaluated. Formal reasoning is not. That gap is large enough, consistent enough, and scientifically meaningful enough to deserve dedicated measurement in future work.

Software and Data

QuantumLean-Bench, including all benchmark problems, Lean formalizations, prompt templates, and manual evaluation annotations, is publicly available at https://github.com/lean-dojo/QuantumLean-Bench. Problems drawn from public datasets and course materials are used in accordance with their respective licenses, which are detailed in the repository README.

Acknowledgements

Robert Joseph George is supported by the DARPA expMath and NSF.2522494 (AIMing) grant. Anima Anandkumar is supported in part by the Bren endowed chair, ONR (MURI grant N00014-18-12624), DARPA expMath and the 2050 senior fellow program at Schmidt Sciences.

Impact Statement

This work contributes an evaluation resource for reasoning and formalization in scientific AI systems. The primary benefit is improved measurement: a paired benchmark can reveal when models that appear strong in natural-language settings still fail to produce faithful formal artifacts. Potential risks arise if benchmark scores are over-interpreted as evidence of autonomous scientific reliability. We therefore view QuantumLean-Bench as a diagnostic benchmark rather than as validation that current models can be trusted without human oversight in high-stakes scientific or engineering workflows.

References

Qm7, 2017. URL https://quantum-machine.org/datasets/.

Every eval ever, 2026. URL https://evalevalai.com/projects/every-eval-ever/.

Qiskit, 2026. URL https://www.ibm.com/quantum/qiskit.

Ai-driven formal theorem proving in the lean ecosystem. https://leandojo.org/, 2026a. Accessed: 2026-02-04.

Lean programming language. https://lean-lang.org/, 2026b. Accessed: 2026-02-04.

Mit opencourseware, 2026. URL https://ocw.mit.edu/.

Basit, A., Shao, M., Asif, M. H., Innan, N., Kashif, M., Marchisio, A., and Shafique, M. Qhackbench: Benchmarking large language models for quantum code generation using pennylane hackathon challenges, 2025. URL https://arxiv.org/abs/2506.20008.

Bohacek, M., Scherrer, N., Dufour, N., Leung, T., Bregler, C., and Chan, S. C. Y. Uncovering competency gaps in large language models and their benchmarks, 2025. URL https://arxiv.org/abs/2512.20638.

Bowman, S. R. Eight things to know about large language models, 2023. URL https://arxiv.org/abs/2304.00612.

Breen, B., Tredici, M. D., McCarran, J., Mijares, J. A., Yin, W. W., Sulimany, K., Taylor, J. M., Koppens, F. H. L., and Englund, D. Ax-prover: A deep reasoning agentic framework for theorem proving in mathematics and quantum physics, 2025. URL https://arxiv.org/abs/2510.12787.

Corporation, M. Quantum katas. URL https://github.com/microsoft/QuantumKatas.

Douglas, M. R., Hoback, S., Mei, A., and Nissim, R. Formalization of qft, 2026. URL https://arxiv.org/abs/2603.15770.

Guo, X., Wang, M., and Zhao, J. Quanbench: Benchmarking quantum code generation with large language models, 2025. URL https://arxiv.org/abs/2510.16779.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Krishna, J. Qiskit quantum katas, 2023. URL https://github.com/jeevesh2002/QuantumKatasQiskit.

Kumarappan, A., Tiwari, M., Song, P., George, R. J., Xiao, C., and Anandkumar, A. Leanagent: Lifelong learning for formal theorem proving, 2025. URL https://arxiv.org/abs/2410.06209.

Li, A., Stein, S., Krishnamoorthy, S., and Ang, J. Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation. *ACM Transactions on Quantum Computing*, 4(2), February 2023. doi: 10.1145/3550488. URL https://doi.org/10.1145/3550488.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K., Orr, L., Zheng, L., Yuksekgonul, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S. M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., and Koreeda, Y. Holistic evaluation of language models, 2023a. URL https://arxiv.org/abs/2211.09110.

Liang, Z., Cheng, J., Yang, R., Ren, H., Song, Z., Wu, D., Qian, X., Li, T., and Shi, Y. Unleashing the potential of llms for quantum computing: A study in quantum architecture design, 2023b. URL https://arxiv.org/abs/2307.08191.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. URL https://arxiv.org/abs/2303.16634.

Mesman, K., Al-Ars, Z., and Möller, M. Qpack: Quantum approximate optimization algorithms as universal benchmark for quantum computers, 2022. URL https://arxiv.org/abs/2103.17193.

Mikuriya, T., Ishigaki, T., Kawarada, M., Minami, S., Kadowaki, T., Suzuki, Y., Naito, S., Takata, S., Kato, T., Basseda, T., Yamada, R., and Takamura, H. Qcoder benchmark: Bridging language generation and quantum hardware through simulator-based feedback, 2025. URL https://arxiv.org/abs/2510.26101.

Minami, S., Ishigaki, T., Hamamura, I., Mikuriya, T., Ma, Y., Okazaki, N., Takamura, H., Suzuki, Y., and Kadowaki, T. Quantumbench: A benchmark for quantum problem solving, 2025. URL https://arxiv.org/abs/2511.00092.

Polu, S. and Sutskever, I. Generative language modeling for automated theorem proving, 2020. URL https://arxiv.org/abs/2009.03393.

Rithvik, S. K. Evaluating large language models on quantum mechanics: A comparative study across diverse models and tasks, 2026. URL https://arxiv.org/abs/2602.19006.

Song, P., Yang, K., and Anandkumar, A. Lean copilot: Large language models as copilots for theorem proving in lean, 2025. URL https://arxiv.org/abs/2404.12534.

Tomesh, T., Gokhale, P., Omole, V., Ravi, G. S., Smith, K. N., Viszlai, J., Wu, X.-C., Hardavellas, N., Martonosi, M. R., and Chong, F. T. Supermarq: A scalable quantum benchmark suite, 2022. URL https://arxiv.org/abs/2202.11045.

Vishwakarma, S., Harkins, F., Golecha, S., Bajpe, V. S., Dupuis, N., Buratti, L., Kremer, D., Faro, I., Puri, R., and Cruz-Benito, J. Qiskit humaneval: An evaluation benchmark for quantum code generative models, 2024. URL https://arxiv.org/abs/2406.14712.

Wang, R., Zhang, J., Jia, Y., Pan, R., Diao, S., Pi, R., and Zhang, T. Theoremllama: Transforming general-purpose llms into lean4 experts, 2024. URL https://arxiv.org/abs/2407.03203.

Waqas, A., Ullah, E., Khan, A., Khalil, F., Ozturk, Z. G., Chumbalkar, V., Saeed-Vafa, D., Jameel, Z., Chen, W., Trejo Bittar, H., Dhillon, J., Singh, R. S., Bychkov, A., Parwani, A. V., Bui, M. M., Schabath, M. B., and Rasool, G. Reasoning beyond accuracy: Expert evaluation of large language models in diagnostic pathology. *medRxiv*, 2025. doi: 10.1101/2025.04.11.25325686. URL https://www.medrxiv.org/content/early/2025/04/20/2025.04.11.25325686.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Wu, Z., Huang, S., Zhou, Z., Ying, H., Yuan, Z., Zhang, W., Lin, D., and Chen, K. Internlm2.5-stepprover: Advancing automated theorem proving via critic-guided search, 2025. URL https://arxiv.org/abs/2410.15700.

Xuejun, Y., Zhong, J., Feng, Z., Zhai, P., Yousefzadeh, R., Ng, W. C., Liu, H., Shou, Z., Xiong, J., Zhou, Y., Ong, C. B., Sugiarto, A. J., Zhang, Y., Tai, W. M., Cao, H., Lu, D., Sun, J., Xu, Q., Xin, S., and Li, Z. Mathesis: Towards formal theorem proving from natural languages, 2025. URL https://arxiv.org/abs/2506.07047.

Yang, K., Swope, A. M., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. Leandojo: Theorem proving with retrieval-augmented language models, 2023. URL https://arxiv.org/abs/2306.15626.

Yousefzadeh, R. and Cao, X. Advocate for complete benchmarks for formal reasoning with formal/informal statements and formal/informal proofs, 2025. URL https://arxiv.org/abs/2507.04719.

Yu, Z., Peng, R., Ding, K., Li, Y., Peng, Z., Liu, M., Zhang, Y., Yuan, Z., Xin, H., Huang, W., Wen, Y., Zhang, G., and Liu, W. Formalmath: Benchmarking formal mathematical reasoning of large language models, 2025. URL https://arxiv.org/abs/2505.02735.

Zhang, H., Wang, R., Pan, R., Wang, W., Meng, B., and Zhang, T. Physprover: Advancing automatic theorem proving for physics, 2026. URL https://arxiv.org/abs/2601.15737.

Zhang, Y., Khan, S., Mahmud, A., et al. Exploring the role of large language models in the scientific method: from hypothesis to discovery. *npj Artificial Intelligence*, 1:14, 2025a. doi: 10.1038/s44387-025-00019-5. URL https://doi.org/10.1038/s44387-025-00019-5.

Zhang, Z., Xu, J., He, Z., Liang, T., Liu, Q., Li, Y., Song, L., Liang, Z., Zhang, Z., Wang, R., Tu, Z., Mi, H., and Yu, D. Deeptheorem: Advancing llm reasoning for theorem proving through natural language and reinforcement

learning, 2025b. URL https://arxiv.org/abs/2505.23754.

Zheng, K., Han, J. M., and Polu, S. Minif2f: a cross-system benchmark for formal olympiad-level mathematics, 2022. URL https://arxiv.org/abs/2109.00110.

A. Data Schema

Each benchmark entry is stored as a single structured record. This design keeps the problem statement, per-model outputs, and manual evaluation metadata aligned under one identifier.

How to read the schema

Problem fields	`id`, `source`, `domain`, `type`, and `problem` define the benchmark item itself.
Response fields	`solution_informal` and `solution_formal` store model outputs for the two response modes.
Evaluation fields	`manual_eval` records the rubric, per-response scores, identified issues, and evaluator notes.

Example dataset entry (truncated)

```
{
  "id": "",
  "source": "",
  "domain": "",
  "type": "",
  "problem": "",
  "metadata": {
    "problem_set": "",
    "question_number": ""
  },
  "citations": [],
  "solution_informal": {
    "deepseek_response": "",
    "qwen3.5-4b_response": "",
    "gemini_response": "",
    "gpt5.4_response": "",
    "stepfun_response": "",
    "stepfun_response_meta": {
      "request_model": "",
      "response_id": "",
      "response_model": "",
      "response_provider": "",
      "created": "",
      "openrouter_model": "",
      "openrouter_provider": "",
      "request_id": ""
    }
  },
  "solution_formal": {
    "qwen3.5-4b_response": "",
    "deepseek_response": "",
    "gpt5.4_response": "",
    "gemini_response": ""
  },
  "manual_eval": {
    "evaluated_at": "",
    "scale": "",
    "rubric": {
      "2": "",
      "1": "",
      "0": ""
    },
    "gold_present": false,
    "responses": {
      "solution_informal.deepseek_response": {
        "score": "",
        "correct": "",
```

```
        "issues": [],
        "notes": ""
      },
      "solution_informal.qwen3.5-4b_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_informal.gemini_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_informal.gpt5.4_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_formal.qwen3.5-4b_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_formal.deepseek_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_formal.gpt5.4_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_formal.gemini_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_informal.stepfun_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      },
      "solution_formal.stepfun_response": {
        "score": "",
        "correct": "",
        "issues": [],
        "notes": ""
      }
    }
  }
}
```

B. Prompt Templates

This appendix summarizes the representative templates used to solicit informal and formal responses. In practice, the placeholders shown below are instantiated with the problem metadata and, for the formal slot, with domain-specific Lean context files.

B.1. Informal Prompt

Input fields

{DOMAIN}	Benchmark domain or subject area.
{{ID}}	Problem identifier.
{{TYPE}}	Task family.
{{PROBLEM}}	Problem statement to solve.

Output contract

- **Only the solution text:** no JSON, no headings such as "Solution:", and no meta-commentary.
- **Reasoning content:** provide a clear derivation or proof sketch with the key steps and conclusion.
- **Information-theoretic prompts:** explain why the relevant state or distribution is independent or indistinguishable whenever that is the core claim.
- **Style:** keep the answer direct rather than narrating the model's thought process.

Template

Role. You are solving a benchmark problem in the domain {DOMAIN}.
Task. Produce an **informal solution** for the benchmark entry below.
Output requirements.

- Return **only** the solution text.
- Include the key derivation or justification needed to support the answer.
- Do not include JSON, section headings, Markdown fences, or meta-commentary.

Problem ID. {{ID}}
Problem type. {{TYPE}}
Problem. {{PROBLEM}}

B.2. Formal Prompt

Problem fields

{DOMAIN}	Benchmark domain or subject area.
{{ID}}	Problem identifier.
{{TYPE}}	Task family.
{{PROBLEM}}	Problem statement to formalize.

Context files

`FORMAL COMMON FILE`	Shared Lean scaffolding file.
`FORMAL CONTEXT FILE`	Domain-specific Lean context file.
`FORMAL EXAMPLES FILE`	Worked Lean examples for the same domain.

Output contract

- **Only Lean 4 code:** no JSON, Markdown fences, headings, or explanatory prose outside the code.
- **Self-contained in principle:** include imports when needed, define helper objects or lemmas, and state assumptions explicitly.
- **No placeholders:** do not use `sorry`, `admit`, vacuous theorem statements, or comments that merely describe missing steps.
- **Faithful core claim:** if the real-world problem is too large to formalize end-to-end, formalize the mathematically checkable core claim that captures the intended solution.

Template

Role. You are solving a benchmark problem in the domain {`DOMAIN`}.
Task. Produce a **formal Lean 4 solution** for the benchmark entry below.
Output requirements.

- Return **only** Lean 4 code.
- Make the solution self-contained enough to typecheck in principle.
- Prefer `import Mathlib`; use PhysLean only when it materially helps.
- State the main claim as a theorem and provide a direct proof structure.

Available context files.

- `FORMAL COMMON FILE` — shared scaffolding.
- `FORMAL CONTEXT FILE` — domain-specific formal context.
- `FORMAL EXAMPLES FILE` — worked examples in the same domain.

Embedded context windows.
```
===== BEGIN SHARED FORMAL CONTEXT =====
FORMAL COMMON
===== END SHARED FORMAL CONTEXT =====
===== BEGIN DOMAIN FORMAL CONTEXT =====
FORMAL CONTEXT
===== END DOMAIN FORMAL CONTEXT =====
===== BEGIN DOMAIN FORMAL EXAMPLES =====
FORMAL EXAMPLES
===== END DOMAIN FORMAL EXAMPLES =====
```
Problem ID. {{`ID`}}
Problem type. {{`TYPE`}}
Problem. {{`PROBLEM`}}

C. Detailed Scoring Rubric

The rubric depends on both (i) **task family** (Numerical / Algorithmic / Proof) and (ii) **response mode** (Informal / Formal). For each combination, we distinguish complete and correct answers from partial progress and from unusable outputs.

C.1. Numerical tasks (Informal)

Score 2 — Complete and correct

- **Final quantity:** outputs the requested number or closed-form expression with the correct sign and scaling.
- **Units:** includes units when the prompt requests them or when omission would affect correctness.
- **Exact-label datasets:** for sources such as QM7, returns the stored target value rather than a qualitative discussion of the representation.

Score 1 — Partial progress

- **Correct setup, incomplete finish:** identifies the correct formula or method but does not reach the final value.
- **Minor arithmetic error:** makes a small numerical slip while otherwise using the correct computation.
- **Missing substitution:** provides the correct expression but omits the numerical substitution explicitly requested in the prompt.

Score 0 — Incorrect or unusable

- **Wrong target:** gives the wrong value or an incorrect formula.
- **No exact answer:** describes the dataset or method without returning the required numerical target.
- **Non-response:** omits the answer, goes off topic, or remains purely qualitative when a quantitative answer is required.

C.2. Numerical tasks (Formal)

Score 2 — Complete and correct

- **Encodes the right target:** represents the requested value or numerical relation as a definition or theorem with the correct mathematical content.
- **Explicit assumptions:** states domain restrictions or non-degeneracy assumptions whenever they matter.
- **Non-vacuous formalization:** the statement is not a tautology such as `True` and does not ignore the problem parameters.

Score 1 — Partial progress

- **Relevant fragment:** provides useful definitions or lemma structure but misses the final target.
- **Incomplete assumptions:** captures the right relation while omitting necessary hypotheses.
- **Proper subset only:** formalizes part of the claim but not the full numerical task requested.

Score 0 — Incorrect or unusable

- **Empty or unrelated code:** the output does not address the task.
- **Vacuous theorem:** uses placeholders or trivial statements that do not encode the requested numerical content.

C.3. Algorithmic tasks (Informal)

Score 2 — Complete and correct

- **Explicit construction:** provides a gate sequence, circuit description, protocol, or code-level algorithm.
- **Constraint satisfaction:** respects any stated restrictions, such as allowed gates or protocol components, or clearly justifies an equivalent transformation.
- **Correctness argument:** explains why the construction works in terms of invariants, state evolution, or input–output behavior.

Score 1 — Partial progress

- **Right idea, incomplete artifact:** identifies the correct method but omits a key gate, measurement, correction, or protocol step.
- **Constraint mismatch:** the construction is broadly relevant but violates an explicit condition in the prompt.
- **Insufficient justification:** gives a plausible construction without enough explanation to establish correctness.

Score 0 — Incorrect or unusable

- **Wrong construction:** the proposed circuit, protocol, or program does not solve the stated task.
- **Pure description only:** discusses the algorithmic idea without giving an explicit artifact.
- **Non-response:** empty or off-topic output.

C.4. Algorithmic tasks (Formal)

Score 2 — Complete and correct

- **Defines the core object:** introduces the circuit, protocol, or program needed by the task.
- **States correctness:** formalizes a property that matches the prompt's intent.
- **Preserves constraints:** includes parameters and restrictions in the statement rather than relegating them to comments.

Score 1 — Partial progress

- **Useful scaffolding:** provides a gate datatype, partial circuit, or helper lemmas without capturing the full construction.
- **Weakened property:** states a related but insufficient correctness condition.
- **Missing constraints:** formalizes the object but not the restrictions that matter for the original problem.

Score 0 — Incorrect or unusable

- **Empty or unrelated output:** the code does not engage with the task.
- **Placeholder formalization:** theorems or definitions are vacuous or ignore the requested algorithmic structure.

C.5. Proof-oriented tasks (Informal)

Score 2 — Complete and correct

- **Correct conclusion:** reaches the requested theorem or claim.
- **Coherent derivation:** includes the key logical steps rather than asserting the result outright.
- **Relevant assumptions:** handles edge cases or assumptions whenever they affect validity.

Score 1 — Partial progress

- **Mostly right but incomplete:** gives the correct intuition while omitting a critical step or justification.
- **Stops short:** develops the argument but does not reach the stated conclusion.
- **Too vague:** identifies the right line of proof without enough detail to count as a full derivation.

Score 0 — Incorrect or unusable

- **Wrong argument or conclusion:** the reasoning does not establish the requested claim.
- **Non-response:** empty or unrelated output.

C.6. Proof-oriented tasks (Formal)

Score 2 — Complete and correct

- **Faithful statement:** expresses the intended claim as a Lean theorem or definition with the right mathematical content.
- **Useful proof structure:** provides a proof skeleton or supporting lemmas that genuinely address the statement.
- **No silent weakening:** avoids vacuity and does not replace the target claim with an easier one.

Score 1 — Partial progress

- **Partial encoding:** formalizes part of the statement or a useful auxiliary lemma.
- **Near miss:** captures the right idea but proves only a nearby weaker claim.
- **Comment-level intent:** indicates the correct direction but fails to encode it as a substantive formal object.

Score 0 — Incorrect or unusable

- **Empty or unrelated code:** no meaningful engagement with the task.
- **Pro-forma proof:** vacuous statements or placeholders that ignore the requested theorem.

D. Prompt and Response Examples

Typesetting note

The examples below are reproduced up to minor whitespace normalization for readability; their substantive content is unchanged.

D.1. Quantum Computing Example

Prompt

Task 1.1. State flip: $|0\rangle \leftrightarrow |1\rangle$.
Input. A qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.
Goal. Change the state of the qubit to $\alpha|1\rangle + \beta|0\rangle$.
Illustrative cases.

- If the qubit is in state $|0\rangle$, change its state to $|1\rangle$.
- If the qubit is in state $|1\rangle$, change its state to $|0\rangle$.
- The operation is self-adjoint: applying it twice returns the qubit to the original state.

Starter code

```
from qiskit import QuantumCircuit

def state_flip(qc: QuantumCircuit, q: int) -> QuantumCircuit:
    """
    Apply a state flip operation to qubit q.
    """
    pass
```

GPT-5.4 informal response

The required transformation is exactly the Pauli X gate. On basis states, $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$, so by linearity

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle.$$

That matches the goal exactly. The implementation is therefore one instruction.

Returned code snippet

```
qc.x(q)
```

GPT-5.4 informal response (continued)

This operation is self-adjoint and involutive, so applying it twice gives the original state back.

GPT-5.4 formal response

```
import Mathlib

namespace QuantumLeanBenchmark.Computing
```

```
def stateFlip : Bool -> Bool := not

theorem stateFlip_zero : stateFlip false = true := by
  rfl

theorem stateFlip_one : stateFlip true = false := by
  rfl

theorem stateFlip_involutive : Function.Involutive stateFlip := by
  intro b
  cases b <;> rfl

end QuantumLeanBenchmark.Computing
```

StepFun informal response

Apply the Pauli-X gate (quantum NOT gate) to the target qubit. The X gate swaps the amplitudes of the $|0\rangle$ and $|1\rangle$ basis states: if the input state is $\alpha|0\rangle + \beta|1\rangle$, applying X yields $\alpha|1\rangle + \beta|0\rangle$. This gate is self-inverse ($X^2 = I$), so applying it twice returns the original state.

Steps.

1. Receive a `QuantumCircuit` object `qc` and an integer `q` (qubit index).
2. Apply `qc.x(q)` to add an X gate on qubit q.
3. Return the modified circuit.

Complexity.

- Time: $O(1)$ — single-gate operation.
- Space: $O(1)$ — constant additional memory; the circuit grows by one gate.

StepFun pseudocode

```
def state_flip(qc, q):
    qc.x(q)
    return qc
```

D.2. Quantum Chemistry Example

Prompt

Given the Coulomb matrix $X \in \mathbb{R}^{23\times 23}$ (see `QuantumLean/BenchmarkData/Chemistry/qm7_problems.json`, id = `qm7_0001`), predict the atomization energy T.
The full matrix is omitted here for brevity.

GPT-5.4 informal response

This is a supervised QM7-style regression benchmark: the Coulomb matrix `X` is an input representation, and the atomization energy `T` is a quantum-chemistry label rather than something one can derive analytically from the matrix by hand without a trained model.
For this benchmark instance, the correct target value is:

$$T = -417.9599914550781.$$

GPT-5.4 formal response

```
-- QM7 benchmark target (atomization energy)
def qm7_0001_T : Float := -417.9599914550781

-- In this dataset, the correct answer is the provided label.
#check qm7_0001_T
```

StepFun informal response

We are given a Coulomb matrix X and asked to predict the atomization energy T. The Coulomb matrix is a representation used in quantum chemistry for machine learning, particularly for predicting molecular properties such as atomization energy. The typical approach is to use a model (for example, kernel ridge regression) trained on a dataset such as QM7. However, no model is provided here; only a single Coulomb matrix is given.
The identifier `qm7_0001` indicates that this instance comes from the QM7 dataset, a benchmark dataset for predicting atomization energies of small organic molecules. QM7 consists of molecules with up to 7 heavy atoms (C, N, O, S, Cl) together with hydrogens, and its Coulomb matrices are typically padded to a fixed size (here 23×23).
Inspecting the matrix, only the first few rows and columns contain non-zero structure, suggesting a small molecule embedded in the padded representation. The diagonal entries are consistent with a Coulomb-matrix construction of the form

$$C_{ii} = \tfrac{1}{2} Z_i^{2.4}, \qquad C_{ij} = \frac{Z_i Z_j}{\|R_i - R_j\|} \quad (i \neq j),$$

which matches standard QM7-style feature definitions.
The first diagonal entry (≈ 36.858) corresponds to a heavier atom (consistent with $Z = 6$ under the $Z^{2.4}$ scaling), and the off-diagonal values (≈ 2.9) are consistent with C–H bond interactions. The remaining diagonal entries of 0.5 correspond to hydrogen atoms ($Z = 1$).
Thus, the matrix corresponds to a molecule with a carbon center and four hydrogens (a methane-like structure embedded in the padded representation). In the QM7 dataset, atomization energies are provided as labels rather than derived from the Coulomb matrix, so the correct target value is the dataset-provided label associated with `qm7_0001`.