SciIF: Benchmarking Scientific Instruction Following Towards Rigorous Scientific Intelligence

Anonymous ACL submission

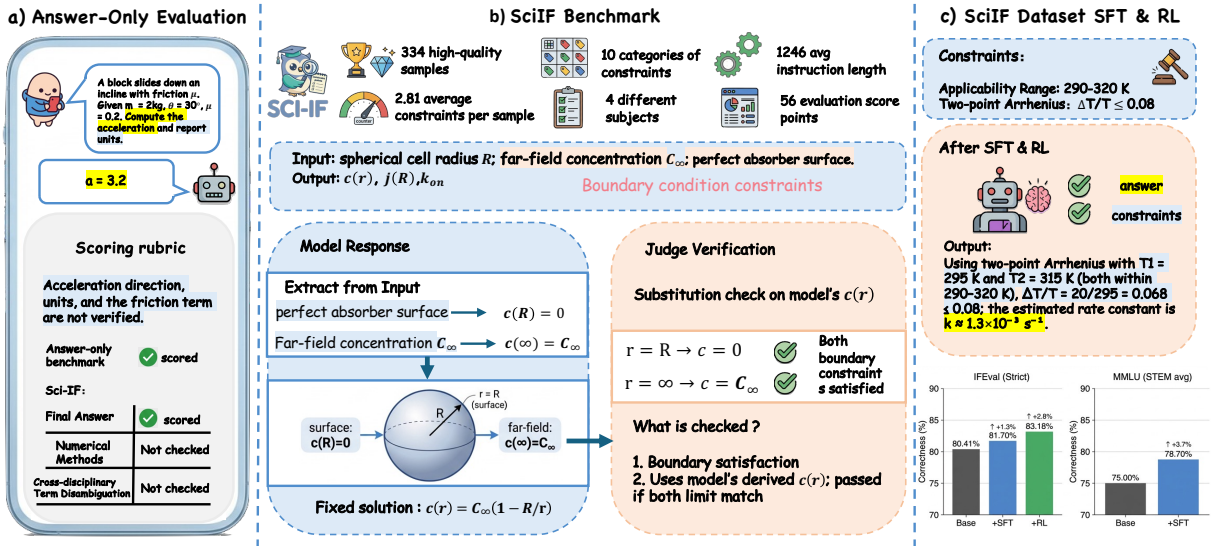

Figure 1: **Overview of the SciIF Benchmark.** a) Existing answer-only scientific benchmarks evaluate only final answer matching, overlooking scientific constraints and instruction compliance. b) SciIF advances this paradigm by introducing explicit scientific constraints and structured verification rubrics, enabling fine-grained assessment of scientific constraint-aware reasoning. c) SciIF provides constraint-grounded SFT and RL datasets, showing that scientific instruction following capability systematically improves correctness, rigor, and instruction adherence.

Abstract

As large language models (LLMs) transition from general knowledge retrieval to complex scientific discovery, their evaluation standards must also incorporate the rigorous norms of scientific inquiry. Existing benchmarks exhibit a critical blind spot: general instruction-following metrics focus on superficial formatting, while domain-specific scientific benchmarks assess only final-answer correctness, often rewarding models that arrive at the right result with the wrong reasons. To address this gap, we introduce scientific instruction following: the capability to solve problems while strictly adhering to the constraints that establish scientific validity. Specifically, we introduce SciIF, a multi-discipline benchmark that evaluates this capability by pairing university-level problems with a fixed catalog of constraints across three pillars: scientific conditions (*e.g.*, boundary checks and assumptions), semantic stability (*e.g.*, unit and symbol conventions), and specific processes (*e.g.*, required numerical methods). Uniquely, SciIF emphasizes auditability, requiring models to provide explicit evidence of constraint satisfaction rather than implicit compliance. By measuring both solution correctness and multi-constraint adherence, SciIF enables fine-grained diagnosis of compositional reasoning failures, ensuring that LLMs can function as reliable agents within the strict logical frameworks of science.

1 Introduction

> *"What we observe is not nature itself, but nature exposed to our method of questioning."*
>
> — Werner Heisenberg

With the rapid adoption of large language mod-

els (LLMs) in scientific discovery, the expectation for model capability is shifting from simple knowledge retrieval to complex problem solving (Hu et al., 2025). However, a fundamental gap exists in how we evaluate these foundation models: in scientific context, the validity of an answer is not intrinsic to the result itself, but is contingent upon whether it is produced within a specific framework of constraints. Unlike everyday user requests where a helpful answer is sufficient, scientific problems are governed by strict norms within the inputs (*e.g.*, assumptions, boundary conditions, definitions, and procedures) that dictate whether a solution is scientifically meaningful or merely numerically plausible. Therefore, as models increasingly tackle rigor-grounded scientific tasks (Mitchener et al., 2025; Wu et al., 2025; Wang et al., 2025b), evaluation must go beyond answer plausibility to assess whether they can strictly adhere to the constraints that establish the scientific validity of that result, namely scientific instruction-following.

However, current benchmarks fail to capture the critical interplay between scientific solutions and justifying constraints. Mainstream instruction-following benchmarks (Zhou et al., 2023; Jiang et al., 2024; Zhang et al., 2025; Qin et al., 2024) primarily emphasize surface-level *format compliance*. For example, IFEval (Zhou et al., 2023) evaluates properties like word counts or JSON validity. While effective for general-purpose dialogue, these criteria are insufficient for science tasks. A model may perfectly satisfy formatting rules yet violating critical scientific rigor, such as applying a formula where it is mathematically undefined or conflating incompatible unit systems. Conversely, prevailing scientific benchmarks (Hendrycks et al., 2021; Wang et al., 2023a) focus almost exclusively on *final-answer correctness*. By treating the reasoning and constraint sanctification process as a black box, these evaluation systematically overlook "right-for-the-wrong-reasons" failures, failing to distinguish a rigorous scientific agent from one that merely arrives at the correct answer without adhering to the underlying scientific framework.

We address this issue by first defining *scientific instruction following*: the ability to solve a scientific problem while explicitly satisfying the constraints that govern correctness, meaning, and process. Unlike general instruction following, which often concerns stylistic preferences and fluency (Wei et al., 2022b), this capability rests on three pillars of scientific validity. First, it requires adherence to *scientific conditions*: models must explicitly check applicability ranges, verify boundary conditions, and state assumptions to prove the solution is valid under the given parameters (Ribeiro et al., 2020). Second, it demands *semantic stability*: models must rigorously adhere to unit conventions, symbol definitions, and terminology standards to prevent meaning drift, where a model implicitly substitutes one quantity for another. Third, it necessitates adherence to specific *scientific processes*: when a specific numerical method or experimental protocol is requested, the model must provide auditable, executable steps rather than a generic conclusion.

In this paper, we introduce SciIF, a benchmark for scientific instruction following across multiple disciplines. Each instance contains: (i) a university-level scientific problem, (ii) an enabled subset of constraints drawn from a predefined catalog, and (iii) a reference solution that is validated to be correct and consistent with the enabled constraints. The catalog is designed around scientific practice and spans three families. *Condition constraints* regulate modeling validity (*e.g.*, assumptions, boundary conditions, applicability range, unit conventions). *Terminology constraints* prevent meaning drift (*e.g.*, cross-disciplinary disambiguation, in-domain term definitions, symbol and constant conventions, variable naming consistency). *Process constraints* require a specific scientific procedure (*e.g.*, numerical methods or experimental methods). Because constraints are drawn from a fixed catalog and combined in controlled ways, SciIF can test compositional instruction following: whether models can coordinate multiple scientific requirements in a single coherent answer (Lake and Baroni, 2018).

A central design goal of SciIF is *auditability*. For each enabled constraint, we specify required evidence that must appear in the model output; judges are instructed not to infer missing evidence. This makes compliance decisions depend on what the model explicitly commits to, rather than on judge guesswork. We report answer correctness, overall multi-constraint compliance, and per-constraint pass rates, enabling fine-grained diagnosis of where scientific instruction following breaks. Furthermore, we demonstrate that fine-tuning on SciIF confers dual benefits, delivering a 2.8% boost on general instruction following (IFE-

val) and a remarkable 8.0% improvement on scientific tasks (MMLU-Physics). This indicates that the rigor required for scientific constraints generalizes effectively, sharpening the model's ability to follow complex rules while deepening its domain expertise.

In summary, our contributions are as follows:

- We introduce SciIF, a multi-discipline benchmark that shifts the evaluation focus from mere numerical correctness to the rigorous execution of scientific instructions.
- We construct a predefined catalog of constraints that supports the generation of controlled problem mixtures for fine-grained failure diagnosis.
- We propose an auditable, evidence-based evaluation protocol that ensures compliance is measured by explicit logical commitments rather than implicit guesswork.
- We demonstrate the utility of SciIF for model alignment, showing that learning strict scientific constraints generalizes to improve both general instruction following and domain-specific reasoning.

2 Related Work

Answer-Centric Scientific Benchmarks. Most scientific and academic evaluations score models primarily by *final-answer correctness*. This includes both *single-discipline* benchmarks such as PhysUniBench (Wang et al., 2025a) that target a specific domain and *multi-discipline* suites—such as MMLU (Hendrycks et al., 2021) and BIG-Bench Hard (Suzgun et al., 2022)—as well as science-focused evaluations such as SciBench (Wang et al., 2023a). While these benchmarks effectively quantify knowledge and reasoning under objective metrics, they often treat the derivation process as a black box. As a result, models can be "right for the wrong reasons": producing a correct value while violating constraints that govern scientific validity and meaning (e.g., unstated assumptions, missing applicability checks, unit inconsistencies, or symbol drift). SciIF complements correctness-centric evaluation by explicitly measuring scientific constraint compliance as a separate axis. Concretely, it scores whether required scientific evidence is *stated and checked* in the output (e.g., explicit boundary substitutions or applicability-range validation), rather than assuming that a plausible derivation implicitly satisfies the rules.

General Instruction Following Evaluation. Instruction following has been studied as cross-task generalization from natural language instructions (Mishra et al., 2022; Sanh et al., 2022; Zhou et al., 2023; Jiang et al., 2024; Zhang et al., 2025; Qin et al., 2024), and improved through instruction tuning and synthetic instruction generation (Wei et al., 2022a; Chung et al., 2022; Wang et al., 2023b). Human-centric evaluation and LLM-as-a-judge frameworks optimize for preference and overall assistant quality (Ouyang et al., 2022; Bai et al., 2022b,a; Dubois et al., 2023; tatsu-lab, 2023; Zheng et al., 2023; The Vicuna Team, 2023), but they typically do not require explicit evidence that scientific constraints are satisfied; fluent outputs can appear acceptable even when scientific conventions are violated. SciIF is complementary: it evaluates whether models both solve the task and explicitly satisfy the scientific constraints that justify the solution, and it enables tracking trade-offs between correctness and constraint compliance under controlled mixtures of constraints.

3 Benchmark Construction

SciIF is designed to evaluate *scientific instruction following* as a capability distinct from answer accuracy. In scientific problem solving, constraints such as unit discipline, symbol meaning, validity conditions, and method requirements are not cosmetic—they determine whether an answer is interpretable, reproducible, and even semantically correct. To surface these failures, SciIF separates evaluation into two axes: (i) correctness of the scientific outcome, and (ii) explicit, auditable compliance with enabled scientific constraints.

3.1 Task Definition

SciIF evaluates scientific instruction following along two separate axes. Given a scientific problem, we score not only whether a model reaches the correct scientific result, but also whether it explicitly follows the enabled scientific constraints.

Each instance is a triple $(x, C, y^\star)$. Here x is a university-level scientific problem, C is a small set of enabled constraints drawn from a fixed catalog, and $y^\star$ is a reference solution that is both correct and compliant with C. A model receives (x, C) and produces an output $\hat{y}$.

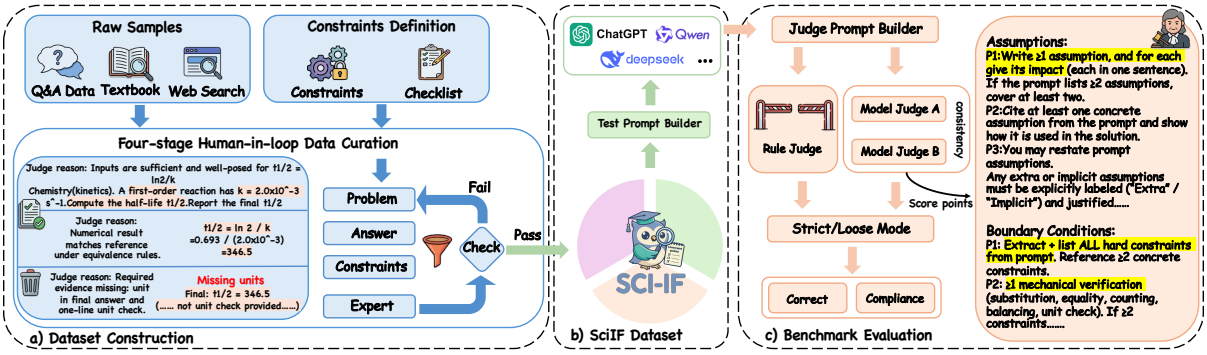

Figure 2: **Overview of SciIF.** We curate scientific QA data from multiple sources and apply a four-stage human-in-the-loop process to produce well-posed problems paired with explicit scientific constraints and auditable evidence checklists. For evaluation, prompt builders generate answer-generation and per-constraint judge prompts, and two independent model judges audit both answer correctness and constraint compliance under Strict or Loose policies. The example highlights a core failure mode: an answer can match the reference numerically yet fail compliance when required evidence such as units and a one-line unit check is missing.

Answer Correctness. We score whether $\hat{y}$ matches $y^\star$ on the required scientific outcome, using task-appropriate equivalence criteria.

Constraint Compliance. In parallel, we score whether $\hat{y}$ provides explicit, problem-grounded evidence for each enabled constraint in C. Judges check evidence presence rather than infer missing assumptions or validity checks, which cleanly separates *correct-but-non-compliant* outputs from *compliant-but-incorrect* ones.

3.2 Dataset and Constraint Catalog

SciIF uses a fixed test set of 334 university-level problems across Biology, Chemistry, Materials, and Physics, and we additionally release 910 training problems used for SFT and verifier-based RL. Each test instance enables a small set of constraints drawn from a fixed catalog of ten atomic constraints. The catalog spans three families: condition constraints (Assumptions, Boundary Conditions, Applicability Range, Units Standard), terminology constraints (Cross-disciplinary Term Disambiguation, Intra-discipline Term Definitions, Symbols & Constants Standardization, Variable Naming Consistency), and process constraints (Numerical Methods, Experimental Methods). Constraints are format-agnostic; we do not require a rigid template. A constraint passes only when the model states explicit, problem-grounded evidence tied to the instances symbols and values, rather than generic boilerplate.

3.3 Construction Pipeline

We construct candidate instances and apply a three-stage quality control pipeline. The stages are designed to separate three failure modes: ill-posed problems, incorrect references, and ungrounded constraints. Only instances that pass all stages are included in the final benchmark.

Stage 1: Problem Validity. Goal: ensure the problem statement itself is well-posed. We check that the prompt provides sufficient information to determine the target quantity, that the scientific setting is internally consistent, and that the enabled constraints are relevant to the task described by the prompt. Instances fail this stage if the question is under-specified, contradictory, or if a constraint is enabled without any anchor in the problem statement that makes it applicable.

Stage 2: Reference Correctness. Goal: ensure the reference solution provides the correct scientific outcome. Independently of constraint evidence, we solve or verify the problem and confirm that $y^\star$ matches the requested target result and key conclusions. Instances fail this stage if the reference contains numerical errors, incorrect reasoning that changes the result, or mismatches the stated targets.

Stage 3: Constraint Grounding and Auditability. Goal: ensure constraints are truly binding and auditable for this instance. We verify that every enabled constraint is instantiated by the prompt and that the reference explicitly provides the required evidence to satisfy that constraint.

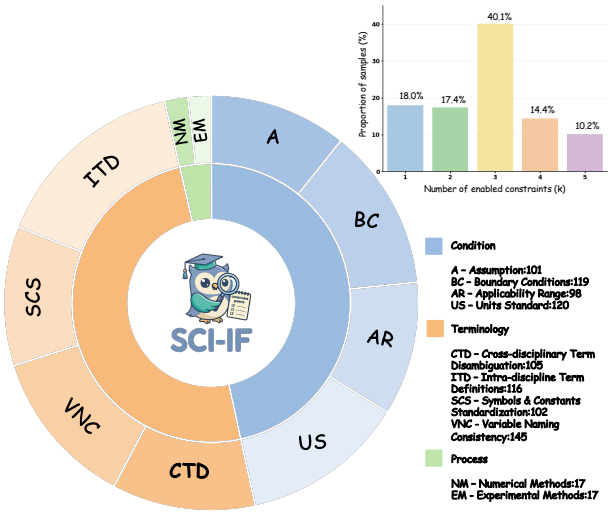

Figure 3: **Constraint composition of SciIF.** Top-right: distribution of the number of enabled constraints per problem. Left: hierarchical breakdown of the constraint catalog, with inner wedges for the three constraint families and outer wedges for the 10 atomic constraints; wedge areas are proportional to how often each constraint is enabled in the test set.

This stage is necessary because correctness alone does not guarantee that a constraint is meaningful or checkable. Instances fail this stage when a constraint is nominal rather than binding, or when the reference does not contain explicit, problem-grounded evidence that would allow a judge to audit compliance.

Stage 4: Domain Expert Review. Goal: ensure the instance reflects realistic scientific practice and remains interpretable to practitioners. After passing automated and internal checks, each candidate is reviewed by a domain expert who assesses (i) whether the scientific context and assumptions are plausible for the stated domain, (ii) whether quantities, units, and parameter ranges are scientifically meaningful, and (iii) whether the reference solution communicates conclusions in a way consistent with field conventions. Instances fail this stage if an expert flags the setting as implausible, identifies domain-knowledge contradictions, or judges that the prompt or reference would mislead a practitioner despite being formally solvable.

3.4 Evaluation and Auditing

SciIF follows a *generate–then–audit* protocol. Given a problem and its enabled constraint set, the model first produces an answer $\hat{y}$. We then audit **answer correctness** and **constraint compliance** as two separate axes, so that being numerically correct can be distinguished from being scientifically auditable.

Constraint Auditing and Aggregation. For each enabled constraint $c \in C$, the auditor assigns PASS or FAIL by checking whether the model output contains **explicit, instance-grounded evidence** for that constraint. Evidence must be clearly stated and tied to the concrete symbols and values in the problem. If evidence is missing, vague, or contradicted by the solution, the constraint fails. For items with multiple constraints, we aggregate with a logical AND over enabled constraints: an item is compliant only if every $c \in C$ passes. This makes failures directly attributable at the constraint level.

Correctness as an Independent Axis. Correctness is audited independently by comparing $\hat{y}$ to the reference $y^\star$ under task-appropriate equivalence criteria for the final outcome and key conclusions. The correctness audit ignores constraint-following issues to avoid conflating scientific reporting failures with arithmetic or reasoning errors. This separation isolates two common regimes in scientific QA: correct-but-non-compliant outputs and compliant-but-incorrect outputs.

3.5 Hybrid Auditing

Rules First, Judges as Fallback. To maximize reproducibility and minimize subjectivity, we use a hybrid auditing pipeline that prioritizes deterministic checks. We first apply a rule-based auditor for signals that admit stable verification, including parsing LaTeX math expressions, checking numerical equivalence under tolerances, validating required unit declarations, and matching constraint-specific evidence patterns. When a rule-based check is not applicable or yields an unreliable decision, we fall back to model-based judging under a strict evidence standard.

Dual Judging with Aligned Standards. Model-based judging uses two independent judges, GPT-5.1 and Gemini-3-Flash, with a conservative agreement rule: a constraint passes only if both judges mark it PASS. In practice, different judges can apply systematically different strictness even on the same response. We therefore align judging standards using a held-out calibration set that is never used for reported results. We identify recurring disagreement modes and encode the corresponding decision thresholds into a fixed, standardized

Model	Condition					Terminology					Process		
	A	BC	AR	US	Avg.	CTD	ITD	SCS	VNC	Avg.	NM	EM	Avg.
Closed-source models													
GPT-5.2 (OpenAI, 2025b)	65.3%	**74.8%**	**32.7%**	38.3%	53.2%	52.4%	**81.9%**	41.2%	**73.1%**	**63.7%**	82.4%	94.1%	**88.3%**
GPT-5.1 (OpenAI, 2025a)	**71.3%**	63.0%	30.6%	**45.8%**	**53.4%**	54.3%	74.1%	47.1%	72.4%	62.8%	70.6%	88.2%	79.4%
GPT-4o (OpenAI, 2024b)	1.0%	0.0%	2.0%	0.0%	0.7%	2.9%	0.0%	4.9%	10.3%	4.8%	0.0%	0.0%	0.0%
GPT-o3 (OpenAI, 2025c)	36.6%	46.2%	15.3%	19.2%	30.0%	29.5%	41.4%	27.5%	38.6%	34.7%	58.8%	76.5%	67.7%
GPT-o4mini (OpenAI, 2024a)	42.6%	38.7%	12.2%	19.2%	28.6%	28.6%	25.0%	18.6%	40.0%	29.0%	76.5%	82.4%	79.5%
Gemini-3 (Google, 2025)	52.5%	52.9%	23.5%	20.8%	38.0%	53.8%	65.5%	47.1%	69.0%	59.7%	**88.2%**	82.4%	85.3%
Grok-4 (xAI, 2025)	43.6%	53.8%	15.3%	33.3%	37.3%	**70.5%**	59.5%	**53.9%**	60.7%	61.0%	58.8%	**100.0%**	79.4%
Claude-4.5Sonnet (Anthropic, 2025)	39.6%	47.1%	26.5%	25.8%	35.0%	36.2%	42.2%	35.3%	49.0%	41.5%	52.9%	82.4%	67.7%
Qwen3-Max (Team, 2025c)	40.6%	47.9%	15.3%	18.3%	31.0%	25.7%	44.0%	21.6%	55.2%	38.0%	82.4%	64.7%	73.6%
Minimax-M2 (MiniMax, 2025)	18.8%	33.6%	5.1%	18.3%	19.5%	15.2%	16.4%	9.8%	33.1%	19.8%	52.9%	58.8%	55.9%
Kimi-K2 (Team, 2025b)	31.7%	37.8%	17.3%	23.3%	27.8%	25.7%	31.9%	26.5%	45.5%	33.4%	**88.2%**	58.8%	73.5%
GLM-4.7 (AI), 2025)	41.6%	36.1%	17.3%	15.8%	27.8%	43.8%	44.8%	26.5%	53.1%	42.8%	58.8%	**100.0%**	79.4%
Open-source models													
Qwen3-235b (Team, 2025d)	24.8%	35.3%	9.2%	**17.5%**	22.1%	15.2%	20.7%	15.7%	31.7%	21.6%	**76.5%**	**64.7%**	**70.6%**
Qwen3-80b (Team, 2025d)	25.7%	**36.1%**	11.2%	8.3%	20.5%	**17.1%**	10.3%	**24.5%**	31.7%	21.2%	35.3%	58.8%	47.1%
Deepseek-v3.2 (Team, 2025a)	**31.7%**	34.5%	**12.2%**	11.7%	**22.8%**	**17.1%**	**21.6%**	20.6%	**34.5%**	**24.0%**	64.7%	**64.7%**	64.7%

Table 1: Per-constraint correctness under Strict mode. **Abbrev.**: A=Assumptions; BC=Boundary Conditions; AR=Applicability Range; US=Units Standard; CTD=Cross-disciplinary Term Disambiguation; ITD=Intra-discipline Term Definitions; SCS=Symbols & Constants Standardization; VNC=Variable Naming Consistency; NM=Numerical Methods; EM=Experimental Methods.

judging prompt. The standardized prompt instructs judges to evaluate evidence presence and consistency only, and to avoid reconstructing missing checks, assumptions, or derivations. After calibration, the judging prompts are fixed for all experiments.

4 Experiments

We evaluate whether scientific QA models can be *both* correct and scientifically auditable. SciIF reports two metrics per model: **answer correctness** and **multi-constraint compliance**.

4.1 Setup

We evaluate on the 334-item SciIF test set spanning Biology, Chemistry, Materials, and Physics. We include a mix of frontier and open models, with the full list deferred to Appendix D.1. All models use identical decoding settings and the same prompting template (Appendix A). Compliance and correctness are scored with our dual-judge auditing protocol.

Post-Training Variants. To test whether SciIF training improves both scientific QA and scientific auditability, we post-train Qwen3-8B on our 910-item training set in two stages. **SFT** targets stronger scientific problem solving and higher answer correctness. We then apply **verifier-based RL** that rewards explicit, problem-grounded constraint evidence, encouraging the model to proactively surface domain-relevant validity information such as assumptions, applicability limits, unit discipline, and boundary checks. We evaluate the base model, SciIF-SFT, and SciIF-RL on SciIF and on external benchmarks (Appendix C).

4.2 Correctness is not Compliance

Our central finding is a persistent gap between getting the right answer and meeting scientific constraints. Across all evaluated models, **answer correctness is substantially higher than strict multi-constraint compliance**. Even strong models exceed 80% correctness, yet overall multi-constraint pass remains below 30% (best: 29.6%). This pattern holds across model families, indicating a systematic failure mode: models often reach the correct final result while omitting explicit evidence for one or more constraints that govern validity, meaning, and reproducibility.

We furthre visualize this gap in Figure 4. For most models, single-constraint pass is relatively strong, which indicates that many scientific requirements are individually achievable when they are the only thing to remember. The failure appears when constraints must be coordinated: multi-constraint overall pass collapses to a much lower regime even when correctness stays high. This separation shows that the dominant error is not inability to solve the underlying scientific problem, but inability to reliably produce a solution as a reproducible artifact with explicit, problem-grounded scientific discipline. In practice, this means an accuracy-only evaluation can label an output as solved even when it omits units, leaves assumptions implicit, fails to state validity limits,

Model	Compliance↑	Correctness↑	IFEval (Strict)↑	MMLU-Bio↑	MMLU-Chem↑	MMLU-Phys↑	MMLU Avg↑
qwen3-8B (Base)	2.1%	24.3%	80.4%	61.0%	**61.0%**	68.0%	63.3%
w/ IFEval (SFT)	—	—	—	43.0% (-18.0%)	37.0% (-24.0%)	61.0% (-7.0%)	47.0% (-16.3%)
w/ SciIF (SFT)	3.1% (+1.0%)	30.2% (+5.9%)	81.7% (+1.3%)	**67.0%** (+6.0%)	58.0% (-3.0%)	**76.0%** (+8.0%)	**67.0%** (+3.7%)
w/ SciIF (SFT+RL)	**7.0%** (+4.9%)	**31.7%** (+7.4%)	**83.2%** (+2.8%)	—	—	—	—

Table 2: Transfer effects beyond SciIF. IFEval (Strict) reports pass rate. MMLU reports accuracy on three STEM subjects and their macro-average.

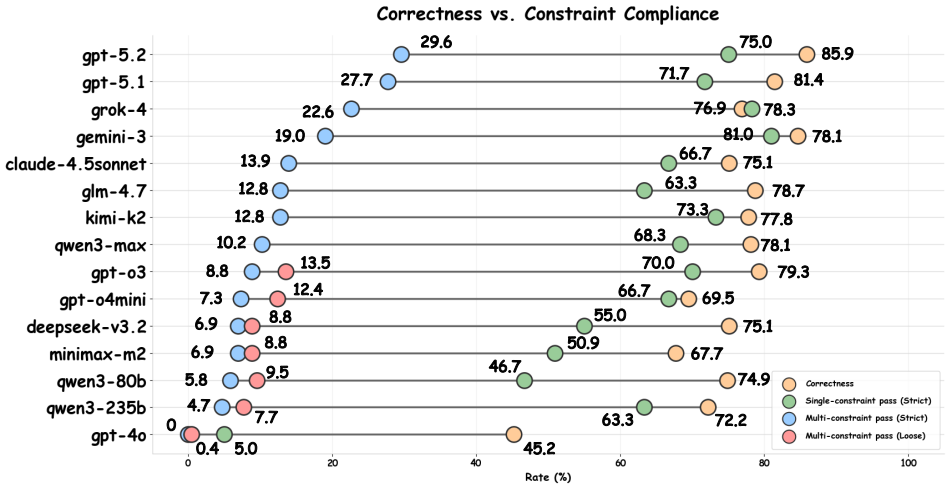

Figure 4: Comparison of three metrics: answer correctness, single-constraint pass, and multi-constraint overall pass. Single-constraint pass is computed on the 60 items with exactly one enabled constraint. Multi-constraint overall pass is computed on the 274 items with multiple enabled constraints, where an item passes only if all enabled constraints pass.

or drifts in symbol meaningissues that can silently break interpretation or downstream reuse.

4.3 Compositional Effects

Correctness and compliance are related but asymmetric. Compliant answers are typically correct, while correct answers are frequently non-compliant, so accuracy-only evaluation overestimates scientific instruction-following ability. We also observe a consistent **compositional collapse**: compliance drops sharply as the number of enabled constraints increases from $k{=}2$ to $k{=}5$ (Appendix D.2), revealing a coordination bottleneck in maintaining multiple scientific requirements within one derivation.

4.4 Generalizability

We test whether training on SciIF data improves performance outside SciIF. Starting from Qwen3-8B, we compare the base model to an SciIF-SFT checkpoint and an SciIF-RL checkpoint. Table 2 shows consistent gains on IFEval (Strict) (Appendix B.6) and improved performance on MMLU science subjects (Appendix C.7 Fig. 8), suggesting that constraint-oriented post-training can transfer to broader instruction adherence and scientific QA rather than only improving in-domain compliance (Appendix A.1).

4.5 Where Models Break

Constraint-level analysis shows that failures concentrate on global semantic requirements such as Units Standard and Applicability Range, which demand explicit, problem-grounded statements and penalize silent assumptions. Full per-constraint breakdowns are reported in Appendix D.

4.6 Explicit rubrics reduce judge variance

Because our evaluation uses LLM-as-a-judge, the judge prompt effectively defines the scoring rule. When constraints are under-specified, different judge models apply different implicit thresholds (e.g., treating 0.1 vs. 0.01 error as "correct"), leading to non-comparable compliance scores. We therefore make scoring points and decision thresholds explicit in the prompt (Appendix A.4). This calibration sharply reduces inter-judge discrepancy: before clarification, GPT-5.1 vs. Gemini-3-Flash yields 68.7% vs. 88.0%; after clarification, 79.1% vs. 81.4%, reducing the gap from 19.3 to 2.3 points. This motivates treating prompt-level rubric specification as a first-class part of protocol.

4.7 Compositional Collapse: Patterns and Anomalies across Models

Beyond aggregate scores, the per-k breakdown reveals a consistent *coordination bottleneck*. As the number of enabled constraints increases, compliance drops steeply for nearly all models, even when answer correctness remains comparatively stable. Figure 5 visualizes this effect for ten representative models selected to cover typical and atypical curve shapes.

A shared baseline shape: smooth exponential-like decay. Strong closed models such as GPT-5.2 and GPT-5.1 exhibit a similar high-start curve at k=2, followed by a near-monotonic decline through k=5. This pattern suggests that failures are rarely caused by a single missing skill. Instead, they arise from *global constraint scheduling*: maintaining unit discipline, symbol meaning, and required validity checks while completing the derivation.

Cliff-drop regimes: fragile constraint composition in open models. Several models enter a "cliff" regime where compliance collapses rapidly once $k \geq 3$. For example, Qwen3-235b reaches low but non-zero compliance at k=2 and k=3, then drops to 0% for $k \geq 4$. This shape indicates that the model can sometimes satisfy isolated constraints, but fails to keep multiple constraints active across a longer solution trajectory.

Non-monotonic anomalies: brittleness and measurement effects. A small set of models exhibit non-monotonic behavior, including partial rebounds at k=5, or "zero-then-recovery" patterns. We treat these as diagnostics rather than evidence of improved compositional ability. Two factors can produce such anomalies: (i) small-sample variance at high k due to fewer items, and (ii) discrete rubric thresholds where a model occasionally "remembers" to add a missing evidence sentence that flips an AND-aggregated decision from FAIL to PASS. These anomalies reinforce a key point: compliance failures are often driven by *missing explicit evidence*, not necessarily by incorrect scientific reasoning.

4.8 Takeaway

SciIF makes visible a gap that correctness-centric benchmarks systematically miss: **scientific correctness and scientific auditability are distinct capabilities**. Current models often produce correct results without reliably producing the explicit evidence needed for reproducible scientific reasoning, and this weakness becomes sharper under multi-constraint composition.

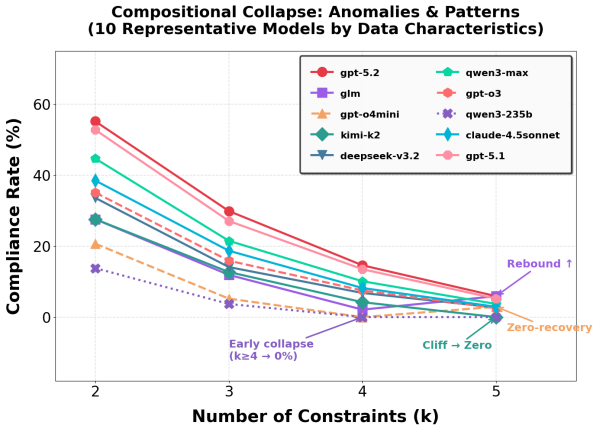

Figure 5: Compositional collapse under increasing constraint load: strict compliance rate versus the number of enabled constraints k for ten representative models.

5 Conclusion

We introduce SciIF to evaluate scientific instruction following, addressing the limitations of current benchmarks that overlook the interplay between solution correctness and constraint satisfaction. Through a taxonomy of domain-specific constraints and an auditable evaluation protocol, SciIF distinguishes between rigorous reasoning and lucky guessing. In addition, our protocol emphasizes transparency and reproducibility, enabling consistent comparisons across models and settings. Crucially, we show that the rigor demanded by SciIF generalizes: models fine-tuned on our data exhibit dual improvements, boosting IFEval by 2.8 points and MMLU-Physics by 8.0 points. We will release SciIF to facilitate the development of LLMs that are not only knowledgeable but strictly compliant with the norms of scientific practice.

Limitation

We acknowledge that our dataset focuses on text-based problems, omitting multimodal scientific tasks (e.g., DNA, RNA sequence generation/understanding). Future work will extend SciIF to multimodal settings and explore iterative, multi-turn scientific agents.

References

Z.AI (Zhipu AI). 2025. *GLM-4.7 Overview*. Z.AI.

Anthropic. 2025. Claude sonnet 4.5. https://www.anthropic.com/claude/sonnet. Official product page.

Yuntao Bai and 1 others. 2022a. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Yuntao Bai and 1 others. 2022b. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Hyung Won Chung and 1 others. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*.

Google. 2025. A new era of intelligence with Gemini 3. https://blog.google/products/gemini/gemini-3/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Ming Hu, Chenglong Ma, Wei Li, Wanghan Xu, Jiamin Wu, Jucheng Hu, Tianbin Li, Guohang Zhuang, Jiaqi Liu, Yingzhou Lu, and 1 others. 2025. A survey of scientific large language models: From data foundations to agent frontiers. *arXiv preprint arXiv:2508.21148*.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. Followbench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

MiniMax. 2025. Minimax m2.1: Significantly enhanced multi-language programming, built for real-world complex tasks. https://www.minimax.io/news/minimax-m21.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C Landsness, Daniel L Barabasi, Siddharth Narayanan, Nicky Evans, and 1 others. 2025. Kosmos: An ai scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*.

OpenAI. 2024a. GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/.

OpenAI. 2024b. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2025a. GPT-5.1: A smarter, more conversational ChatGPT. https://openai.com/index/gpt-5-1/.

OpenAI. 2025b. Introducing GPT-5.2. https://openai.com/index/introducing-gpt-5-2/.

OpenAI. 2025c. Introducing openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/. Published 2025-04-16. Accessed 2026-01-06.

Long Ouyang and 1 others. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Marco Ribeiro and 1 others. 2020. Beyond accuracy: Behavioral testing of nlp models. In *ACL*.

Victor Sanh and 1 others. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

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

tatsu-lab. 2023. Alpacaeval: An automatic evaluator of instruction-following language models. https://github.com/tatsu-lab/alpaca_eval. Accessed 2025-12-22.

Deepseek Team. 2025a. Deepseek-v3.2: Model overview and capabilities.

Moonshot AI Team. 2025b. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Qwen Team. 2025c. Qwen3-max our largest and most capable model to date. https://qwen.ai/blog?id=qwen3-max. Official Qwen blog post.

Qwen Team. 2025d. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

The Vicuna Team. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90% chatgpt quality. https://lmsys.org/articles/vicuna-an-open-source-chatbot-impressing-gpt-4-with-90-chatgpt-quality. Accessed 2025-12-22.

Lintao Wang, Encheng Su, Jiaqi Liu, Pengze Li, Peng Xia, Jiabei Xiao, Wenlong Zhang, Xinnan Dai, Xi Chen, Yuan Meng, Mingyu Ding, Lei Bai, Wanli Ouyang, Shixiang Tang, Aoran Wang, and Xinzhu Ma. 2025a. Physunibench: An undergraduate-level physics reasoning benchmark for multimodal models. *Preprint*, arXiv:2506.17667.

Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2023a. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yizhou Wang, Chen Tang, Han Deng, Jiabei Xiao, Jiaqi Liu, Jianyu Wu, Jun Yao, Pengze Li, Encheng Su, Lintao Wang, and 1 others. 2025b. Scireasoner: Laying the scientific reasoning ground across disciplines. *arXiv preprint arXiv:2509.21320*.

Jason Wei, Maarten Bosma, and 1 others. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei and 1 others. 2022b. Emergent abilities of large language models. *arXiv preprint*.

Hengkui Wu, Liujiang Liu, Jihua He, Qihao Wang, Keke Zhao, Shuyang Hu, Renle Fu, Dahao Liang, Lingyu Zeng, Bruce Liu, and 1 others. 2025. Bigbang-proton technical report: Next-word-prediction is scientific multitask learner. *arXiv preprint arXiv:2510.00129*.

xAI. 2025. Grok 4. https://x.ai/news/grok-4.

Tao Zhang, Chenglin Zhu, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, and 1 others. 2025. Cfbench: A comprehensive constraints-following benchmark for llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32926–32944.

Lianmin Zheng and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Appendix Contents

A Evaluation Protocol and Prompt Templates

This appendix provides the information needed to reproduce our evaluation and training signals. We first document the generate–then–audit protocol and the rule-based verifier–judge pipeline, then list the answer-generation prompts, judge prompts, and decision policies used in Strict and Loose modes.

A.1 Generate–then–audit protocol

Two-axis evaluation. Each model output is evaluated on two axes: answer correctness and constraint compliance. Correctness checks whether the required scientific outcome matches the reference under task-appropriate equivalence criteria. Compliance checks whether the output contains explicit, problem-grounded evidence for each enabled constraint. Judges evaluate evidence presence rather than infer missing checks, assumptions, or definitions.

Per-constraint auditing and aggregation. Each enabled constraint is audited independently and receives PASS/FAIL. For multi-constraint items, item-level compliance is the logical AND across enabled constraints.

Dual-judge robustness. We use two independent judges and apply a two-of-two rule. A constraint passes only if both judges return PASS, reducing sensitivity to single-judge drift and stylistic preference.

A.2 Rule-based verifier with judge fallback

We prioritize deterministic checks when feasible and fall back to LLM judges when a rule-based check is not applicable or fails to parse the output. Rule-based checks include numerical equivalence under task-defined criteria, expression parsing when needed, unit and dimensional consistency checks when units are stated, and mechanical substitution checks for boundary or limiting cases. When a rule cannot be executed reliably, the decision is delegated to the judge prompts below.

A.3 Answer generation prompts

A.1 Strict Mode: Answer Generation Prompt

```
You are solving a university-level
    scientific problem.

PROBLEM (verbatim):
[RAW_QUESTION_WITH_CONSTRAINTS]

Enabled constraints:
[CONSTRAINT_NAME_LIST]

Required evidence (MUST appear;
    missing any item => non-compliant)
    :
[CONSTRAINT_SPECIFIC_EVIDENCE_POINTS]

Writing rules:
```

```
- Integrate evidence into the
    solution narrative with natural
    transitions.
- Do not use checklist-style rubric
    blocks or fixed labels.
- Do not claim compliance with
    slogans; show explicit, auditable
    statements and checks.
- Do not use markdown tables for
    definitions; define symbols inline
     in sentences.
- Follow the problem's stated units
    and precision requirements.

Answer format:
- Short plan
- Derivation / reasoning (equations
    if needed)
- Final Answer (clearly stated, with
    units if applicable)

Answer:
```

A.4 Judge prompts

We audit correctness and constraint compliance separately. For compliance, judges evaluate one enabled constraint at a time against explicit evidence points and return a pointwise verdict.

A.3 Strict Mode: Per-Constraint Judge Prompt

```
You are an expert validator for
    university-level [SUBJECT]
    problems.

Constraint name:
[CONSTRAINT_NAME]

Constraint description:
[CONSTRAINT_DESCRIPTION]

Required evidence points (evaluate
    EACH independently):
POINT_1: [POINT_1_DESC]
POINT_2: [POINT_2_DESC]
...

Problem:
[QUESTION]

Reference answer (if provided):
[GOLD_ANSWER]

Model answer:
[MODEL_ANSWER]

Decision policy:
- Decide whether each point has
    explicit, problem-grounded
    evidence in the model answer.
- Do NOT infer missing evidence.
- If the model claims a check, verify
     it is correct and consistent with
     its own work.

Output format (MUST follow exactly):
POINT_1: YES or NO [brief reason if
    NO]
POINT_2: YES or NO [brief reason if
    NO]
...
OVERALL: YES or NO
OVERALL_REASON: [<=150 chars, list
    failed points]
```

A.4 Loose Mode: Per-Constraint Judge Prompt

```
You are an expert validator for
    university-level [SUBJECT]
    problems.

Constraint name:
[CONSTRAINT_NAME]

Constraint description:
[CONSTRAINT_DESCRIPTION]

Required evidence points (evaluate
    EACH independently):
POINT_1 [MAIN]: [POINT_1_DESC]
POINT_2 [MAIN]: [POINT_2_DESC]
POINT_3 [SECONDARY]: [POINT_3_DESC]
...

Problem:
[QUESTION]

Reference answer (if provided):
[GOLD_ANSWER]

Model answer:
[MODEL_ANSWER]

Decision policy:
- Same as STRICT for pointwise
    evidence, non-inference, and
    truthfulness checks.
- OVERALL is YES only if all MAIN
    points are YES. SECONDARY points
    may be NO.

Output format (MUST follow exactly):
POINT_1: YES or NO [brief reason if
    NO]
POINT_2: YES or NO [brief reason if
    NO]
POINT_3: YES or NO [brief reason if
    NO]
...
OVERALL: YES or NO
OVERALL_REASON: [<=150 chars, list
    failed MAIN points]
```

A.5 Decision policies

Strict policy. A point is YES only when the model output contains explicit evidence that is problem-grounded, correct, and consistent with

Aspect	Strict	Loose
Overall verdict	All points must pass	All main points must pass
Missing evidence	Always FAIL	Always FAIL
Semantic checks	Applied broadly	Focused on main points
Use case	Main leaderboard, auditable scoring	Diagnosis of omission vs. violation

Table 3: High-level differences between Strict and Loose compliance modes.

the model's own derivation. Missing evidence is NO. Incorrect claimed checks are NO.

Loose policy. Loose mode keeps the same non-inference and truthfulness requirements, but only MAIN points are required for overall PASS. SECONDARY points may fail without failing the constraint.

A.6 Strict vs. Loose summary

We use two compliance modes to separate missing-evidence failures from substantive violations. Strict is the primary reporting setting and is designed for auditability. Loose is a diagnostic setting used to test whether low scores are driven mainly by omission of write-ups.

B Benchmark Details

This section provides additional information about rubric structure, instance construction, and judge calibration.We provide a high-level overview of the ten constraints in Table 4 to clarify what constitutes auditable evidence at a glance.The full point-wise definitions, edge cases, and the Strict/Loose point split are provided in the supplementary rubric specification.

B.1 Source-type composition of the 1244-item pool

Our full data pool contains 1244 items and is intentionally dominated by textbook-style problems. We use web-derived items as a secondary source to broaden coverage, while keeping QA-style items as a smaller component. Overall, the source-type mixture follows an approximate 7:2:1 ratio: about 70% textbook, about 20% web, and about 10% QA. This design emphasizes academically grounded problem structures while still injecting topical diversity and alternative phrasing patterns.

B.2 Evidence-based rubrics

Each enabled constraint is accompanied by evidence points designed to be checkable without reconstruction or guesswork. A constraint passes only if all required points pass. Any missing, vague, or contradictory evidence causes failure. We release the full point-level rubric specification in machine-readable form as supplementary material.

B.3 Optional regeneration and stability filtering

For a small subset of items, generation can yield good problems but brittle references. When needed, we regenerate reference candidates and retain instances that fall within a target stability window, filtering out instances that are effectively trivial or near-impossible under the same constraints.

B.4 Judge scale alignment

Two-of-two voting reduces noise but does not guarantee that judges apply identical evidence thresholds. We align judge scales using a held-out calibration set that is never used for reported scores. We compare judge outputs per evidence point, identify recurring disagreement patterns, and refine judge prompts by making those thresholds explicit and executable. After calibration, prompts are fixed for all experiments.

B.5 Case study: judge disagreement and prompt calibration

We conducted a targeted case study to quantify and reduce judge arbitrariness. We sampled 50 instances and re-audited the same model outputs with two judge models, then compared pointwise PASS/FAIL decisions under the same rubric. We observed non-trivial disagreement: 19 out of 50 instances had at least one constraint-level mismatch, with 20 constraint-level mismatches in total. Disagreements concentrated on globally semantic constraints such as SYMBOLS & CONSTANTS STANDARDIZATION and INTRA-DISCIPLINE DEFINITIONS, where the effective threshold for "explicit evidence" differed across judges. These findings motivated prompt calibration that removes packaging-dependent expectations and rewrites evidence points into problem-grounded, executable checks, leaving minimal room for stylistic interpretation.

Constraint	Family	Main evidence required (high level)
Assumptions	Condition	State at least one key assumption used, explain its impact on the method, and anchor it to a concrete step in this solution.
Boundary Conditions	Condition	Extract the boundary or hard conditions from the prompt and provide at least one mechanical verification that the derived solution satisfies them.
Applicability Range	Condition	Name the approximation or model used, state an explicit validity range, and describe a failure mode outside the range with a correct direction or trend.
Units Standard	Condition	Give units for key variables at first use, keep units consistent, and include a short dimensional or unit self-check tied to the target quantity.
Cross-disciplinary Disambiguation	Terminology	Identify ambiguous terms, state the intended meaning and a non-intended meaning, and point to where the intended meaning is used in the solution.
Intra-discipline Definitions	Terminology	Provide a plain definition and a formal criterion, then apply the criterion in one concrete step of the solution.
Symbols & Constants Standardization	Terminology	Define symbols used in the final result, declare constants with units and source when required, and avoid symbol drift.
Variable Naming Consistency	Terminology	Maintain one symbol per quantity throughout the solution and prevent semantic drift in symbol meaning and units.
Numerical Methods	Process	Name the algorithm, provide the update rule using problem symbols, and show at least one instantiated numerical iteration.
Experimental Methods	Process	Provide an executable procedure tied to problem variables, include auditable anchors, and state how uncertainty affects the target quantity.

Table 4: High-level rubric summary for the ten constraints. Detailed point-wise definitions and decision policies are in Appendix A.3–A.4.

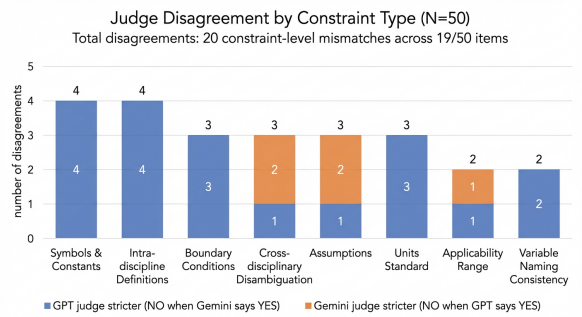

Figure 6: Judge disagreement by constraint type on a 50-item audit set. Bars count constraint-level mismatches between two judges, split by which judge is stricter. Across the 50 items, we observe 20 constraint-level mismatches spanning 19 items, with disagreements concentrated in SYMBOLS & CONSTANTS and INTRA-DISCIPLINE DEFINITIONS.

B.6 External consistency check on instruction following

As an external sanity check, we evaluated Qwen3-8B and its post-trained variants on IFEval under Strict and Loose settings. Verifier-guided reinforcement learning improves instruction following beyond supervised fine-tuning alone, with the RL variant achieving higher Strict performance than both the base and SFT variants. This supports the interpretation that constraint-oriented post-training strengthens general instruction discipline rather than only improving in-domain compliance on SciIF.

B.7 Constraint rubric summary

We define ten atomic constraints grouped into three families. Table 4 summarizes the main evidence requirements at a high level.

C Post-Training Details

This section documents the SFT and verifier-based RL objectives, reward construction, and implementation notes.

C.1 SFT objective

We train a LoRA adapter on top of Qwen3-8B with the standard next-token likelihood objective:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; -\sum_{t} \log \pi_\theta\big(y_t^* \mid q, y_{<t}^*\big), \quad (1)$$

where y^* is the reference solution.

C.2 RL objective

We optimize a KL-regularized RL objective:

$$J(\pi_\theta) = \mathbb{E}_{(q,r)\sim\mathcal{D}}\Big[\mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\big[R(q,o,r)\big] - \beta\, D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid q) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid q)\big)\Big] \quad (2)$$

where π_{ref} is initialized from the SFT checkpoint and β controls deviation from the reference.

C.3 Constraint verifier

The verifier evaluates a response against the enabled constraints:

$$V : (q, o, r) \mapsto \mathbf{v} \in \{0, 1\}^d, \tag{3}$$

where $v_i = 1$ iff the output provides the required evidence to satisfy constraint r_i.

C.4 Reward construction with grouped severity

Let $s_i \in \{0, 1\}$ denote PASS/FAIL for constraint r_i from the verifier. We partition constraints into groups $g \in \mathcal{G}$, assign within-group weights w_i, and between-group weights W_g.

For a group g, the normalized group score is:

$$R_g = \frac{\sum_{i \in g} w_i \, s_i}{\sum_{i \in g} w_i}. \tag{4}$$

We combine group scores into an overall constraint-compliance score:

$$R_{\mathrm{c}}(q, o, r) = \frac{\sum_{g \in \mathcal{G}} W_g \, R_g}{\sum_{g \in \mathcal{G}} W_g}. \tag{5}$$

We add a binary answer-correctness signal $R_{\mathrm{a}}(q, o)$ and compute the final scalar reward:

$$R(q, o, r) = 0.7 \, R_{\mathrm{c}}(q, o, r) + 0.3 \, R_{\mathrm{a}}(q, o). \tag{6}$$

C.5 Implementation notes and selected results

We perform PPO-style optimization with KL regularization to the SFT reference policy. On our evaluation, RL increases the single-constraint pass rate from 5.0% to 11.7% and answer correctness from 24.3% to 25.7%. Among individual constraints, symbol and constant conventions improve from 5.9% to 24.5%.

C.6 Case study: what verifier-based RL changes on IFEval

To understand what our verifier-based RL stage changes beyond in-domain compliance on SciIF, we run an external consistency check on IFEval under the Strict setting. We compare Qwen3-8B in a zero-shot setting against Qwen3-8B-RL. The RL variant improves Strict accuracy from 80.41% (435/541) to 83.18% (450/541), a gain of 2.77 points corresponding to 15 additional passing instances. To characterize the behavioral shift, we focus on the 48 cases where RL passes while zero-shot fails.

Where the gains concentrate. The RL gains are not uniformly distributed across instruction types. They cluster in constraints that penalize extra "helpful" text and require precise surface-form control. The most frequent improved categories are sentence and word length constraints, case transformation constraints, forbidden keyword constraints, letter-frequency constraints, and punctuation bans. This pattern indicates that RL primarily reduces a consistent failure mode of general chat-style models: they often prefer conversational packaging and elaboration over strict compliance when the instruction demands a tight output envelope.

Behavioral shift: from conversational packaging to executable compliance. Across improved cases, we observe three recurring corrections. First, the RL model reduces preambles and meta-commentary that violate strict formatting requirements. Second, it exhibits tighter control of quantitative length constraints, often showing implicit self-monitoring behavior such as stopping early and compressing content while preserving task intent. Third, it better coordinates multiple constraints simultaneously, avoiding partial satisfaction where one constraint is met but another is silently violated.

Representative examples. We present three representative Strict-mode examples that illustrate the dominant error patterns. In all examples, the task content is easy for both models. The failures arise from instruction-following discipline rather than missing knowledge.

Example A: strict JSON-only output

Instruction. "Make an advertisement for a new diaper product. The entire output must be JSON format."
Zero-shot failure. The model adds an explanatory preamble and wraps the JSON in a Markdown code block. This violates the requirement that the *entire* output be valid JSON.
RL success. The model outputs raw JSON directly with no surrounding text.
Takeaway. RL suppresses the "helpful assistant" habit of adding extra text that breaks a strict output envelope.

Example B: word-count constraint

Instruction. "Write a short blog post about a trip to Japan using less than 300 words."
Zero-shot failure. The model produces a coherent post but exceeds the word limit substantially. The failure is not semantic but quantitative. It reflects weak internal length control during generation.

RL success. The RL model stays under the limit and preserves narrative coherence. In several improved instances, the RL model also shows self-verification behavior, such as ending early and compressing details while keeping the post well-formed.
Takeaway. RL improves quantitative constraint control without requiring external tools, suggesting the policy learns to budget output length as part of instruction-following.

Example C: multi-constraint coordination

Instruction. "Write a tweet without using capital letters. Include at least four hashtags starting with #."
Zero-shot failure. The model satisfies the lowercase constraint but adds extra explanatory text and quotation-style packaging. It also risks failing the implied "tweet-only" output expectation in strict instruction-following evaluation.
RL success. The model outputs a tweet-like text directly, meets the lowercase constraint, and includes four or more hashtags.
Takeaway. RL improves coordination across multiple simultaneous constraints and reduces "partial compliance" where one constraint is met but the output format drifts into meta-commentary.

Implication for our training objective. These results support an interpretation consistent with our SciIF findings. Verifier-based RL strengthens a general notion of instruction discipline, especially in constraints that demand explicit surface-form control. This aligns with our benchmark design, where compliance is judged from written evidence. The IFEval case study suggests that the same training signal that improves auditable scientific constraint satisfaction also reduces format and length violations in a different instruction-following domain.

C.7 Case study: SciIF SFT improves structured reasoning on MMLU

We include a concrete MMLU-style example to illustrate a qualitative change we repeatedly observe after SciIF SFT 8: the model becomes more stable during solution writing and more reliable at making the final discrete decision that the task demands. In this item, all variants know the correct physics formula, yet they differ sharply in execution discipline. The SciIF SFT variant keeps exponent arithmetic explicit, performs a quick order-of-magnitude sanity check, and maps the estimate to the closest option without drifting into irrelevant text or misreading the scale.

Model	$k=2$	$k=3$	$k=4$	$k=5$
GPT-5.2	55.2%	29.9%	14.6%	5.9%
GPT-5.1	52.8%	27.1%	13.5%	5.2%
Gemini-3	49.3%	23.4%	11.7%	4.3%
Qwen3-Max	44.7%	21.5%	10.1%	3.7%
Grok-4	43.8%	22.2%	9.4%	3.1%
Claude-4.5Sonnet	38.5%	18.7%	8.2%	2.9%
GPT-o3	35.1%	15.9%	7.5%	2.5%
Deepseek-v3.2	33.6%	14.1%	6.8%	2.7%
Minimax-M2	31.7%	13.3%	6.2%	2.0%
Kimi-K2	27.6%	12.7%	4.2%	0.0%
GLM-4.7	27.6%	11.9%	2.1%	5.9%
GPT-o4mini	20.7%	5.2%	0.0%	2.9%
Qwen3-80b	17.2%	4.5%	0.0%	0.0%
Qwen3-235b	13.8%	3.7%	0.0%	0.0%
GPT-4o	3.3%	1.2%	0.0%	0.0%
Qwen3-8b	0.0%	0.0%	0.0%	0.0%

Table 5: Multi-constraint compliance rate (%) as a function of the number of enabled constraints k. Compliance drops rapidly as k increases, showing a compositional collapse effect: models that perform well on few constraints often fail to coordinate multiple scientific requirements simultaneously.

D Additional Experimental Results and Diagnostics

This section reports extended tables and diagnostic analyses that complement the main paper.

D.1 Models and inference settings

We evaluate: GPT-5.2, GPT-5.1, Gemini-3, Grok-4, Qwen3-max, GPT-o3, GPT-o4mini, Claude-4.5sonnet, Deepseek-v3.2, Minimax-M2, and GPT-4o. All models use identical inference settings: temperature $= 0$, max tokens $= 4096$, no tools or web.

D.2 Compositional collapse vs. number of constraints

Compliance drops sharply as the number of enabled constraints increases. We report the per-k breakdown in Table 5.

E AI Assistant Use

We used GPT and Gemini for code refactoring/optimization and for polishing the manuscripts language and formatting. All changes were reviewed by the authors, who take full responsibility for the final content and results.

F Human Validation Protocol for Equivalence Between Model and Reference Answers

This appendix explains, for human readers, how we manually assess whether a models answer is

equivalent to the reference (gold) answer. The process targets university-level problems and focuses on the equivalence of numerical results, key conclusions, and core reasoning, while keeping style or compliance issues out of scope.

1. What We Check

- **Numerical agreement:** When a numerical result is required, we verify that the final numbers match the reference. To ensure comparability, all reported numbers use exactly four significant figures; if the reference specifies a range or tolerance, we adhere to that.
- **Conclusion agreement:** Categorical outcomes (true/false, multiple-choice selection, sign/direction, inequality relations) must match the reference. If the reference states "undetermined" or "requires more information," any definite conclusion in the model answer is considered non-equivalent.
- **Conceptual and reasoning equivalence:** Different wording is acceptable, but the core method or theorem employed should be equivalent (e.g., the same physical law or the same convergence criterion). Alternative methods are acceptable if they are logically equivalent for this problem and properly justified.

2. What We Do Not Penalize

- Minor formatting or wording differences, different paragraphing, or non-essential elaborations that do not affect the substance of the answer.

3. Step-by-Step Procedure

1. **Preparation:** Review the problem and the reference answer; extract key data, conditions, and the final conclusion.
2. **Evidence tagging:** Read the model answer and highlight explicit, problem-specific evidence (data, formulas, method statements), ensuring it ties to the symbols/conditions of this problem.
3. **Numerical check:** Compare all numerical conclusions item by item, reporting numbers with exactly four significant figures and verifying units and sign/direction where applicable.
4. **Conclusion check:** Verify that selections, truth values, and relational/directional statements match the reference; if the reference is "indeterminate," ensure the model maintains the same stance.
5. **Reasoning check:** Confirm that the core concepts and methods align with the reference. If a different method is used, verify that it is logically equivalent and correct for this problem.
6. **Final decision:** If numerical results, key conclusions, and core reasoning align, we mark the answers as equivalent. Otherwise, we record the discrepancy with a brief reason (e.g., "different final value," "method not equivalent," "definite conclusion given where reference is indeterminate").

4. Boundaries and Notes

- This process does not evaluate formatting templates, submission style, or other compliance requirements (handled separately).
- When the reference specifies units or measurement conventions, we follow them; if not specified, we use the conventions implied by the problem statement.
- For readability, we integrate evidence within a natural narrative rather than relying on checklists.

Using this protocol, we provide transparent, reproducible, and academically rigorous judgments of equivalence between model answers and the reference solutions.

Pre-calibration ambiguity example: Biology-48, Symbols & Constants Standardization

Observed mismatch. Two judges disagreed on whether the model output satisfies SYMBOLS & CONSTANTS STANDARDIZATION. Both judges accepted the computations, but they differed on whether S_1 and S_2 must be explicitly declared with units rather than treated as instances of a generic substrate symbol.

Constraint under audit. SYMBOLS & CONSTANTS STANDARDIZATION

Evidence points (operationalization).

- **POINT_1 Symbol declaration:** all symbols used in computation are explicitly declared with units at least once.
- **POINT_2 Constant declaration:** any constant used provides numeric value, unit, and standard source when required.
- **POINT_3 No drift:** symbols do not change meaning or units across the solution.

Pre-calibration judge outputs (pointwise).

Calibration change applied. We revised the evidence wording to remove packaging dependence and eliminate judge-specific degrees of freedom. The updated requirement is model-independent and executable: for every constant or symbol that appears in computation, the answer must state its meaning and unit at least once, either inline or in a short declaration sentence. We clarified that a dedicated symbol table is not required, and that a generic symbol declaration does not automatically cover indexed instances unless the indexing relation is stated.

Figure 7: A concrete pre-calibration disagreement case. The same model output is accepted by one judge and rejected by the other due to different implicit thresholds for symbol explicitness. Prompt and rubric calibration rewrites evidence points into packaging-neutral, executable requirements to reduce judge-specific interpretation space.

MMLU case study: High-school Physics 0119 (order-of-magnitude choice)

Problem (excerpt). The mass of the Earth is 5.97×10^{24} kg. The Moon has mass 7.35×10^{22} kg. The distance between centers is 3.84×10^{8} m. Estimate the gravitational force of the Earth on the Moon. Options: A 10^{39} N, B 10^{29} N, C 10^{19} N, D 10^{9} N.
Gold choice: C.

Predictions (same input item).

MMLU-750: Pred. A × (Incorrect)
MMLU-750 + IFEval-50: Pred. D × (Incorrect)

MMLU-750 + SCI-IF-50: Pred. C ✓ (Correct)

Key excerpts and failure modes.

MMLU-750

"`Answer: C.`"
`(then repetitive looping and unrelated continuation)`

`final extracted option drifts to A`

MMLU-750 + IFEval-50

`Uses` $F = G\frac{Mm}{r^2}$ `and computes an estimate, then states:`
$F \approx 2 \times 10^{20}$ `N, so the closest option is D (`10^{9} `N)`.

`Correct magnitude, wrong discrete mapping`

MMLU-750 + SCI-IF-50

`Writes exponent arithmetic explicitly:`
$Mm \approx (10^{24})(10^{22}) = 10^{46}$, $G \approx 10^{-11}$, $r^2 \approx (10^8)^2 = 10^{16}$.
`So` $F \approx 10^{-11} \cdot 10^{46}/10^{16} = 10^{19-20}$ `N.`

`Selects C (`10^{19} `N) and adds a brief sanity check on scale.`

Takeaway. This example isolates a common post-training effect: SciIF SFT does not merely encourage longer solutions. It improves decision discipline at the end of the chain by making intermediate scaling steps explicit, reducing irrelevant generation drift, and enforcing a final, task-grounded mapping from estimate to choice.

Figure 8: A representative MMLU-style item where SciIF SFT improves both solution stability and the final multiple-choice decision. All variants know the correct formula, but they differ in execution discipline: the SciIF SFT variant makes order-of-magnitude reasoning explicit and selects the closest option correctly, while the IFEval variant mis-maps the scale and the baseline exhibits generation drift.