

Right Answer, Wrong Method: Shortcut Hacking Misleads the Evaluation of LLM Reasoning on Frontier Science Benchmarks

Xuan Ren^{1,*}, Weiqi Zhai^{1,*}, Tianle Pu^{1,*}, Yihua Zhu¹, Hu Wei^{1,†}, Bing Zhao^{2,†}

¹Alibaba Group

²Alibaba DAMO Academy

*These authors contributed equally., †Corresponding authors.

Abstract

Scientific reasoning benchmarks typically evaluate large language models (LLMs) using final-answer accuracy. However, a correct answer does not necessarily demonstrate the reasoning capability targeted by the problem. We identify Solution Hacking, a failure mode in which an LLM reaches the correct answer through invalid shortcuts, such as numerical search, enumeration, guessing, or answer-first verification, without providing a valid task-targeted derivation. We systematically analyze this phenomenon across difficulty levels, scientific domains, and frontier models. Solution hacking increases sharply with benchmark difficulty, from 2.2% on common problems to 28.3% on Olympiad-level problems and 37.4% on HLE. Moreover, 8.2%-44.1% of answers credited as correct across frontier models are identified as hacked solutions. We further develop expert-inspired anti-hacking strategies, including an automatic judge and a test-time instruction. The results show that suppressing shortcut behavior substantially reduces reported accuracy while having a smaller effect on correct and non-hacked accuracy. These findings reveal that answer-only evaluation can overestimate the scientific reasoning capabilities of frontier LLMs.

1 Introduction

Scientific reasoning has become a central capability for large language models (LLMs), driving the emergence of increasingly difficult frontier benchmarks[1–4]. These benchmarks typically treat final-answer accuracy as the gold standard for evaluating model performance, with higher scores widely regarded as evidence of stronger reasoning ability. Yet the reasoning process is equally important, as it determines whether the final answer is supported by a valid step-by-step reasoning process. However, evaluating a solution is far more difficult, especially for challenging frontier scientific problems. There are methods that attempt to verify the reasoning process of a solution, but they often overlook a key fact: whether the solution actually puts the target reasoning ability into practice.

Unfortunately, overlooking this fact will mislead the evaluation of the true reasoning capabilities of frontier LLMs. To better understand this concern, we conduct a motivating analysis using problems sampled from three benchmark tiers of increasing difficulty: common mathematical reasoning, Olympiad-level problems, and HLE. We then ask frontier LLMs to solve each problem independently. Surprisingly, we find that LLMs often bypass the target reasoning ability by adopting a cheaper but invalid shortcut that may recover the correct answer without providing a valid, task-targeted solution process. Here, take solving a quadratic equation for

example, as shown in Figure 1. We expect an LLM to derive the exact roots using the quadratic formula. In practice, however, it may simply enumerate numerical values until it identifies candidate roots that satisfy the equation. More importantly, this behavior occurs more frequently as benchmark difficulty increases.

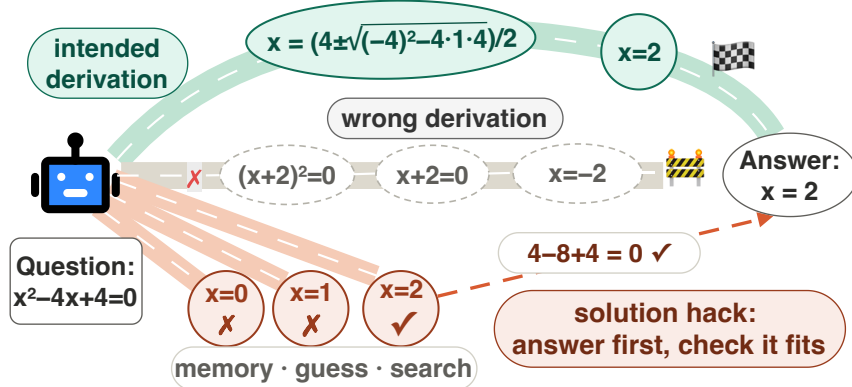


Figure 1 Three solution paths for the same quadratic-equation problem. The upper path represents the expected reasoning process, in which the LLM exercises the targeted reasoning capability and derives the correct answer. The middle path represents a reasoning error, where an invalid derivation leads to an incorrect answer. The lower path represents Solution Hacking, where the LLM obtains the correct answer through enumeration, guessing, search, or answer-first verification without completing the targeted derivation. Traditional evaluation can distinguish the upper and middle paths, but may fail to distinguish the upper and lower paths because both yield the correct final answer.

To further illustrate this phenomenon, we define this phenomenon as solution hacking, in which an LLM produces a correct answer through an invalid solution process that bypasses the targeted reasoning capability. Unlike reasoning errors, solution hacking does not simply refer to the use of a method that differs from the reference solution. Rather, it occurs when a model relies on search, enumeration, guessing, or answer-first verification to obtain or confirm the correct answer without providing a valid derivation that independently establishes its correctness. In addition, we conduct a detailed analysis of solution hacking across problem difficulties, scientific domains, and frontier models. First, solution hacking becomes more frequent as problem difficulty increases, rising from 2.2% on common problems to 28.3% on Olympiad-level problems and 37.4% on HLE. Second, it increases the reported final-answer accuracy of LLMs. Across many frontier models evaluated, such as Claude Opus 4.7, between 8.2% and 44.1% of the answers credited as correct are identified as hacked solutions. Third, weaker models are more likely to choose to solution hacking when they cannot solve a problem accurately, while problems with short and easily verifiable answers are particularly hacking.

Finally, to decrease solution hacking, we further develop anti-hacking strategies based on expert judgment and expert-designed principles. Specifically, we distill how domain experts determine whether a solution genuinely exercises the targeted reasoning capability or instead relies on shortcuts such as search, enumeration, guessing, or answer-first verification. We then translate these principles into two practical tools: an automatic judge for identifying hacked solutions and a test-time instruction that discourages shortcut behavior while allowing the model to abstain when it cannot provide a valid derivation. On a challenging mathematics subset, the strongest anti-hacking instruction reduces reported final-answer accuracy from 50.6% to 37.3% and increases the abstention rate from 0% to 27.3%. In contrast, the accuracy of answers judged to be both correct and non-hacked decreases more modestly, from 41.2% to 35.2%. This result suggests that much of the removed score was previously supported by shortcut strategies rather than successful reasoning, demonstrating that anti-hacking strategies can provide a more faithful evaluation of the scientific reasoning capabilities of frontier LLMs.

Our contributions are summarized as follows:

- We identify and carefully define Solution Hacking, a failure mode in which an LLM reaches a correct answer through an invalid shortcut that bypasses the targeted reasoning capability, and show how this

phenomenon can mislead the evaluation of frontier LLMs.

- We systematically analyze solution hacking across benchmark difficulties, scientific domains, and frontier models, revealing its prevalence, its inflation of reported final-answer accuracy, and the model- and problem-level factors associated with its occurrence.
- We develop expert-inspired anti-hacking strategies, including an automatic judge and a test-time instruction, and show that suppressing shortcut behavior enables a more faithful evaluation of frontier LLM reasoning capabilities.

2 Related Work

Benchmark Evaluation and Shortcut Exploitation. That optimizers exploit misspecified objectives is classical [5–7], and related gaming behaviors are now studied in LLMs [8, 9]. The reliability of benchmark evaluation may also be affected by score inflation from contamination [10, 11], performance collapse under problem perturbations [12, 13], and leaderboard distortions [14]. Shortcut learning provides a broader account of models relying on unintended patterns or strategies rather than the targeted capability [15]. Within scientific reasoning, shortcut-like behaviors have appeared in prior studies, but mostly as isolated symptoms or benchmark-specific concerns. Trial-and-error and proof-by-example are included in taxonomies of proof fallacies [16]; pattern recognition and brute-force enumeration are discussed as motivations for proof grading [17]; and OlymMATH reports heuristic guessing in case studies while explicitly declining to quantify its prevalence [18]. Benchmark designers have also attempted to defend against individual strategies through guess-resistant answer spaces [19], brute-force-defeating perturbations [13], and Lean-verified subsets [18]. In contrast, we study a general evaluation-time failure mode in which an answer-correct solution bypasses the targeted reasoning capability through search, enumeration, guessing, or answer-first verification. We define this behavior as solution hacking and systematically quantify it across scientific domains, benchmark difficulties, and frontier LLMs.

Reasoning Process Evaluation and Capability Verification. Process supervision and step-level evaluation examine whether intermediate reasoning steps are correct, useful, or suitable as training signals [20–22]. Related work further grades complete derivations rather than final answers and documents a substantial answer–rigor gap. Expert grading of contest proofs shows that models with strong final-answer performance may still produce invalid proofs containing broken logic, unjustified assumptions, or fallacious steps [16, 17, 23]. IneqMath reports that top-model accuracy drops substantially when every reasoning step is scrutinized [24], while correct answers in knowledge-graph question answering may rely on unfaithful reasoning chains [25]. These studies primarily ask whether a reasoning process is locally or globally valid. Our work instead asks whether the solution actually exercises the reasoning capability targeted by the benchmark. A hacked solution may contain plausible or locally valid steps, for example, the enumerated candidates may be correctly checked against the stated conditions, and may therefore evade conventional step-level error detection, even though the overall strategy bypasses the targeted capability. To our knowledge, prior work has not jointly defined this behavior, anchored its detection in blinded expert judgments across multiple scientific domains, quantified its impact on reported performance, and investigated practical mitigation strategies. We also differ from general LLM-as-a-judge methods [26–28]: our judge evaluates how a correct answer was obtained rather than whether the answer itself is correct.

3 Solution Hacking

To precisely characterize the evaluation gap introduced above, we first formalize what it means for an answer-correct solution to bypass the reasoning capability targeted by a benchmark. Consider a benchmark item (q, a^*) , where q is a problem and a^* is its reference answer. Given q , an LLM generates a solution s with an extractable final answer $\hat{a}(s)$, which is typically scored by $\mathbb{K}[\hat{a}(s) \equiv a^*]$. Interpreting this score as evidence of scientific reasoning implicitly assumes that an answer-correct solution actually exercises the capability that the item is designed to assess.

Let $\mathcal{C}(q)$ denote the targeted reasoning capability of item q , and let $M(q)$ denote the set of task-appropriate strategies that domain experts accept as legitimate ways of exercising this capability. Importantly, $M(q)$ may

contain multiple valid approaches and is not restricted to the reference solution.

Definition 1 (Solution hacking). A solution s to an item (q, a^*) is a solution hack if $\hat{a}(s) \equiv a^*$, but at an essential step, the solution substitutes a shortcut strategy outside $\overline{M(q)}$ for the targeted reasoning capability $\mathcal{C}(q)$. Although the shortcut may recover or verify the correct answer, it does not provide a valid, task-appropriate derivation that independently establishes the answer.

Definition 1 distinguishes solution hacking from both ordinary reasoning errors and valid alternative solutions. We operationalize the definition through three questions. **(T1) Targeted crux:** does the shortcut replace an essential step that the item is intended to test? Guessing an incidental intermediate value is not sufficient. **(T2) Capability bypass:** does the strategy replace, rather than legitimately exercise, the targeted reasoning capability? A method is not a hack merely because it differs from the reference solution. **(T3) Derivational support:** does the reasoning independently establish the answer, rather than merely show that a generated candidate is compatible with some of the problem constraints?

These criteria make the distinction role-based rather than technique-based. The same surface strategy may be legitimate in one problem but constitute hacking in another, depending on the role it plays in the solution. For example, testing candidate roots is legitimate when trial is an accepted method and the search space is exhaustively covered. In contrast, recalling a candidate answer and checking only that it satisfies one observed condition is a hack when the problem requires that answer to be derived and competing candidates are never ruled out. The definition therefore excludes benign uses of similar techniques, such as checking an already-derived numerical result, citing an accepted theorem as an input, eliminating explicit answer choices, or conducting exhaustive case analysis with justified coverage.

Hack: substitutes the targeted step	Clean: the same technique in a legitimate role
An answer is recalled or guessed and checked against constraints that merely admit it; competing candidates are not ruled out	Candidate testing is the accepted method and the check is conclusive, such as trial roots or undetermined coefficients
A formula is asserted as “known” at the step where it should be derived	An established theorem is cited as an allowed input and then correctly applied
A general conclusion is inferred from a few observed cases	All relevant cases are covered, or the general claim is formally proved
Numerical search replaces an expected analytical derivation	Numerical computation is used only to verify an already-derived result
A stated constraint is silently removed and a simpler problem is solved	An approximation is explicitly stated, justified, and permitted by the task
Orthogonal distinction: an incorrect answer produced by an honest, complete derivation is a reasoning error, whereas a correct answer produced by a substituted shortcut may constitute solution hacking.	

Table 1 Boundary between solution hacking and legitimate reasoning. Classification depends on the role of a strategy relative to the targeted reasoning step, rather than on the surface technique or final-answer correctness.

3.1 A Taxonomy of Hacking Strategies

With the construct defined, we next examine how solution hacking appears in practice. Through manual analysis of audited frontier-model solutions, we identify five recurring strategies and one residual category:

- **Numerical search** ($\mathcal{H}_{\text{num}}$): locating an answer through bisection, Newton iteration, or repeated numerical evaluation when an analytical derivation is expected.
- **Enumeration** ($\mathcal{H}_{\text{enum}}$): searching over a candidate space, including partial enumeration that stops once the first compatible answer is found without establishing completeness.
- **Pattern guessing** ($\mathcal{H}_{\text{pat}}$): extrapolating a general result from a small number of observed cases without justifying the generalization.
- **Formula guessing** ($\mathcal{H}_{\text{form}}$): postulating a functional form from memory, plausibility, or dimensional considerations and then fitting or checking its constants instead of deriving the form.

- **Answer guessing** ($\mathcal{H}_{\text{ans}}$): proposing a plausible answer, often from memory, and checking only that it is consistent with the given constraints without ruling out alternatives.
- **Other shortcuts** ($\mathcal{H}_{\text{oth}}$): other forms of substitution at the targeted step, such as asserting a decisive result without support or silently replacing the original problem with a simpler one.

Manifestations across subjects. Although this taxonomy is initially distilled from mathematical solutions, the underlying construct—substitution at the targeted reasoning step—extends across scientific domains. What changes across subjects is the form of the shortcut. In physics and chemistry, two patterns occur frequently.

The first is memory retrieval followed by consistency checking: the model recalls the crucial formula, object, or compound instead of deriving it, and then uses the provided information only to verify the recalled candidate. Accordingly, 66% of physics hacks are categorized as formula guessing, whereas 60% of chemistry hacks are categorized as answer guessing. The second pattern is problem substitution, in which the model removes a stated constraint or ignores an essential feature before solving the resulting simpler problem. Such cases are included in the residual category and occur more frequently in physics and chemistry than in mathematics (18.7% and 14.5%, compared with 6.7%). We therefore retain the common taxonomy for reporting while interpreting each category according to its domain-specific manifestation (App. K).

3.2 Case Study

The following example illustrates how an answer can be correct even though the reasoning capability targeted by the problem is never exercised.

Mathematics: numerical search for six consecutive primes

The problem asks the model to identify six consecutive primes whose product equals a given 32-digit integer N . The model first estimates their scale by computing " $x^6 \approx N \Rightarrow x \approx 200162.3$; the primes must be near 200162". It then tests odd integers in [200130, 200200] for primality and multiplies candidate sequences until one matches N . This process returns the correct six primes, but it replaces the intended number-theoretic derivation with a localized numerical search.

This example satisfies all three criteria in Definition 1. The numerical search replaces the central reasoning step of the problem, does not exercise the targeted number-theoretic capability, and verifies only that the discovered candidate matches the given product. Nevertheless, because the final answer is correct, conventional answer-only evaluation awards the solution full credit.

Solution hacking may also appear as an explicit fallback policy rather than an accidental mistake. For example, one audited model states, "I will bet on a small solution I missed", before proposing and checking a candidate answer. Additional cases from physics, chemistry, and mathematics are provided in Appendix B, together with a clean-versus-hacked mirror pair in Appendix C.

4 Method

We use blinded expert annotations to build two instruments for solution hacking: a judge that measures whether a completed solution is hacked (our measurement instrument), and an anti-hack answering prompt that discourages such shortcuts at generation time (our mitigation instrument). Both are anchored to the same expert-labeled solutions and to the construct boundary in Table 1.

Throughout, correctness and hacking are evaluated separately. Correctness is a lightweight final-answer equivalence check against the reference answer (numeric answers within 5% relative tolerance; symbolic answers up to algebraic equivalence), computed once and shared across all judges. The hack judge sees only the problem and the model’s solution—never the reference answer or the correctness verdict—so wrong answers are not automatically treated as hacks.

4.1 Metrics

For model M on benchmark B , let $C_i \in \{0, 1\}$ denote final-answer correctness and $H_i \in \{0, 1\}$ a hack (Def. 1):

$$\begin{aligned} \text{Acc} &= \mathbb{E}_i[C_i], \quad \text{HR} = \mathbb{E}_i[H_i], \quad \text{HR}_{|\text{corr}} = \mathbb{E}_i[H_i \mid C_i=1], \\ \text{DAA} &= \mathbb{E}_i[C_i(1 - H_i)]. \end{aligned}$$

DAA (derivation-adjusted accuracy) credits only solutions that are both correct and non-hacked; the score inflation $\Delta_{\text{infl}} = (\text{Acc} - \text{DAA})/\text{Acc} = \text{HR}_{|\text{corr}}$ is the fraction of credited answers that did not demonstrate the intended derivation. To validate the judge, we use agreement with expert hack/clean labels as the primary reliability metric, with Cohen’s κ secondary.

4.2 Build Expert-Anchored Judge

Stage 1: Seed auditor and stratified corpus. We first built a hack-audit prompt with Claude Code (Opus 4.7), iteratively refined by a computer scientist against manually reviewed examples. Given a problem–solution pair, an auditor running this prompt outputs structured JSON: an analysis of the essential steps, a binary hack/clean verdict, a strategy label from our taxonomy (§3.1), and quoted decisive evidence. For corpus auditing we deployed it on Gemini-3.1-Pro-Preview (auditor v2; Table 2, “Stage 1”), already under a no-self-audit rule: Gemini-authored solutions were audited by Claude Opus 4.7 instead. We then had GPT-5.2 [29], Claude Opus 4.7 [30], and Gemini-3.1-Pro-Preview [31] answer hard mathematics, physics, and chemistry problems from HLE [1] and olympiad-level sources, and used the auditor’s verdicts to build a judge-stratified annotation pool containing both flagged and clean solutions—natural hack prevalence is sparse and uneven, and unstratified annotation would yield too few positive cases for calibration.

Stage 2: Blinded expert annotation. From this pool we sampled 300 solutions (101 mathematics, 100 physics, 99 chemistry), spread across the three answering models and stratified by the Stage-1 verdict (107 flagged, 193 clean), then shuffled and blinded. Solutions were labeled by pools of PhD experts in the corresponding field (8 annotators in mathematics, 7 in physics, 6 in chemistry): each solution received a hack/clean label and a strategy category from one expert, following written guidelines (released) that operationalize Table 1, and annotators then performed a second self-review pass over their own labels to catch and correct errors. These 300 labels form the common anchor set (Table 10 in App. J); we split it into 180 development examples for prompt refinement and detector selection and 120 held-out test examples used exactly once. Two findings drive the rest of the paper: experts confirm substantial hacking on frontier items (35.2% overall: 45.9% math, 26.5% physics, 33.0% chemistry), and the Stage-1 auditor already under-counts relative to experts, motivating both calibration and the lower-bound framing.

Detector	Overall	Math	Phys	Chem
Single auditor (Gemini)	71.6	76.5	76.8	61.6
Panel, self-inclusive	77.7	82.2	82.0	68.7
Panel, no-self-audit	75.4	79.6	81.6	64.9

Table 2 Detector–expert agreement (%; pooled dev+test). Bold = deployed detector. Dev 75.7%, test 75.0% (App. F).

Subject	Human HR	Detector HR	Recall
Mathematics	45.9	43.9	0.76
Physics	26.5	22.4	0.58
Chemistry	33.0	30.9	0.44
Overall	35.2	32.4	0.61

Table 3 The detector under-flags in every subject and recovers only 61.2% of expert-confirmed hacks: all reported hack ratios are lower bounds.

Model	Overall				Hack ratio by subject (%)		
	Acc	HR	HR _{corr}	DAA	Math	Physics	Chem
GPT-5.5 [†]	48.1	22.0	21.0	38.0	26	11	31
Kimi-K3 [†]	46.9	12.3	11.0	41.7	12	5	28
DeepSeek-V4-Pro [†]	45.7	22.7	23.9	34.8	28	12	31
Qwen3.7-Max	45.6	17.7	17.8	37.5	22	6	28
Gemini-3.1-Pro-Preview	44.0	14.1	8.2	40.4	20	10	7
Claude Opus 4.7	42.0	20.9	21.2	33.1	28	7	27
GLM-5.2 [†]	37.4	20.5	19.5	30.1	30	8	19
Claude Opus 4.8 [†]	34.0	38.2	23.7	26.0	47	24	43
GPT-5.2	19.6	35.7	19.0	15.8	45	21	40
DeepSeek-V3.2	15.5	50.3	28.8	11.0	61	31	59
GPT-4.1	10.1	60.4	44.1	5.7	71	43	66

Table 4 Main corpus results (3,528 frontier solutions; deployed detector; %). Rows ordered by accuracy: weaker models hack more and inflate more. All HR/DAA entries are lower bounds (Table 3). [†]Models audited in a follow-up run under the identical protocol; Kimi-K3 and GLM-5.2 could not produce a response on 7.2% of items, and DeepSeek-V4-Pro on 22.7% (extreme-length reasoning), so their HR is, if anything, further underestimated. Original-run models were backfilled to near-full item coverage (residual non-response $\leq 2.7\%$ per model).

Stage 3: Calibration and final detector. Using the development anchors, we refined the audit prompt to better handle recurring boundary cases: enumeration is allowed when it is part of the intended method, citing a standard theorem is allowed, and several weak clues can together indicate a hack. To keep the judge comparable to the audited models, we only used peer-level judges: Gemini-3.1-Pro-Preview, Claude Opus 4.7, and GPT-5.2. The final detector is a majority vote of these three judges, each using the prompt revision that best matches expert labels on the development split (the final revision for Gemini and Claude, and the previous one for GPT-5.2, which tends to over-flag under the strictest rules; App. G). To avoid self-grading bias, a model never audits its own solution; in those cases, only the other two judges vote, and both must agree to flag. This slightly lowers raw agreement, but makes model comparisons fairer. The deployed panel reaches 75.4% agreement with experts, up from 71.6% for the Stage-1 auditor (Table 2); agreement is also stable across splits (dev 75.7%, test 75.0%). Further ablations and the self-audit counterfactual are in Appendix F. The detector’s errors are asymmetric on the anchor set: it falsely flags only 17% of expert-clean solutions (32/190), but misses 39% of expert-confirmed hacks (40/103), recovering 61.2% of them overall (Table 3). Since misses outnumber false alarms in every subject, the detector under-flags relative to experts, so all reported hack ratios should be read as conservative lower bounds.

4.3 Build Expert-Anchored Anti-Hack Prompt

The same anchor set is also used to build a generation-time intervention. While the judge asks whether a finished solution replaced an essential derivation with a shortcut, the answering prompt turns the same expert cues into instructions that discourage such shortcuts during generation.

We build four variants that enforce the same boundary in different ways. The ban-list variant explicitly forbids each shortcut strategy in our taxonomy. The necessity variant requires the solution to show that the answer is necessary or unique. The guardrail variant states the grading rule clearly while still allowing legitimate enumeration when that is the intended method. The pre-commit variant asks the model to state its planned derivation before solving and to mark key steps as derived or guessed. All variants also let the model abstain by outputting “CANNOT SOLVE RIGOROUSLY” when it cannot complete a rigorous solution. This matters: without an abstention option, the prompt may encourage the model to hide shortcuts instead of honestly admitting it cannot solve the problem. We evaluate these variants in §5.5. The answering prompt and the judge are built from the same blinded expert labels, so they enforce the same distinction between clean reasoning and shortcut hacking.

5 Experiments

We use the calibrated judge for four purposes: (i) measuring hacking across SOTA models on the frontier corpus; (ii) comparing hack rates across datasets of different difficulty; (iii) testing how much the anti-hack answering prompt reduces hacking, and how model performance changes when it does; and (iv) analyzing what makes the judge itself reliable.

Setup and difficulty tiers. We define three difficulty tiers by problem provenance: easy textbook problems (SciBench [32]; MATH-500 [21, 33] level ≤ 2), medium contest problems (IMO-Bench [34], PHYBench [4], SciOlympiad [35]), and hard frontier problems (HLE [1]). The medium and hard tiers together form our main corpus of 3,528 solutions across the three subjects, produced by eleven models (GPT-5.2, Claude Opus 4.7, Gemini-3.1-Pro-Preview, DeepSeek-V3.2 [36], Qwen3.7-Max [37], GPT-4.1 [38] as a deliberately weaker reference, and, in a follow-up run under the identical protocol, GPT-5.5 [39], Claude Opus 4.8 [40], Kimi-K3 [41], GLM-5.2 [42], and DeepSeek-V4-Pro [43]) under a neutral chain-of-thought prompt that never mentions hacking (further details in App. A).

5.1 Substantial Hacking Persists on Frontier-Level Problems

Table 4 reports the main result on the hard tier, where the overall hack ratio is 33.2%. The key quantity is HR_{corr} , the share of credited answers that were in fact hacked. It grows steeply as models weaken, from 8.2% for Gemini-3.1-Pro-Preview to 44.1% for GPT-4.1, so reported accuracy overstates derived competence by a corresponding margin, and DAA falls below Acc for every model. Because the detector under-counts (Table 3), all of these gaps are lower bounds. Strategy profiles are dominated by formula guessing and answer guessing, matching the two cross-subject mechanisms described in §3.1 (Table 13 and App. K).

5.2 Hack Rates Rise on Harder Benchmark Sets

Using the same deployed detector, we compare hack rates across benchmark sets of different difficulty. On the easy tier, the hack ratio is only 2.2%, while accuracy remains high at 89.1% (Figure 2; Table 5, App. A). The ratio rises to 28.3% on competition problems and 37.4% on frontier HLE items, and the same broad pattern appears in every subject (Figure 2). Overall, hacking is rare on the easy tier but much more common on harder benchmarks, which is less consistent with indiscriminate flagging by the judge. One possible reason this phenomenon has received limited attention is that solutions in the easy tier are accessible to a much wider audience, whereas medium- and hard-tier problems often require domain expertise beyond what many computer science practitioners can readily verify at the derivation level.

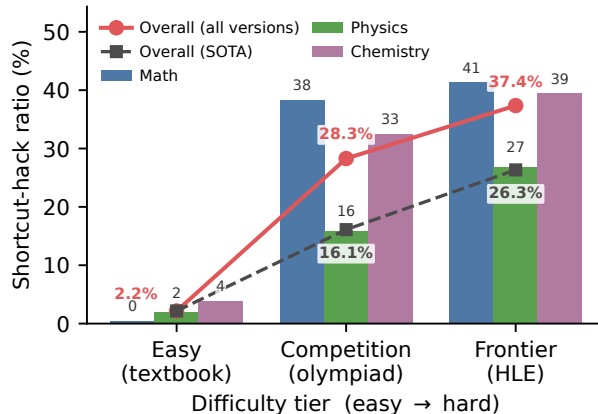


Figure 2 Hack ratio by difficulty tier and subject (deployed detector): the rise is monotone in every subject, $2.2 \rightarrow 28.3 \rightarrow 37.4\%$ pooled over all model versions (solid line; bars give the per-subject pooled values). The dashed line pools only the strongest current version of each family ($2.2 \rightarrow 16.1 \rightarrow 26.3\%$). Competition/frontier tiers pool the eleven corpus models; the easy tier uses the SOTA model set.

5.3 Searchability and When Hacking Pays Off

We highlight two patterns that may be relevant to hacking. First, hack rates vary sharply across benchmarks, from 16% on PHYBench to 42% on HLE-Math, and this ordering does not simply follow difficulty: PHYBench is among the hardest benchmarks, yet is hacked least. One possible explanation is that some benchmarks have answers that are easier to guess and check than others. In particular, integer and identification targets may be easier to search over than symbolic-formula targets (Figure 6), though we treat this as a suggestive pattern rather than a causal claim. Second, model capability appears relevant in a different way: weaker models hack more often when they fail to solve a problem cleanly, but hacks from stronger models are more likely to end in answers that are counted as correct. In that sense, hacking seems to pay off more for stronger models (Figure 3; App. A).

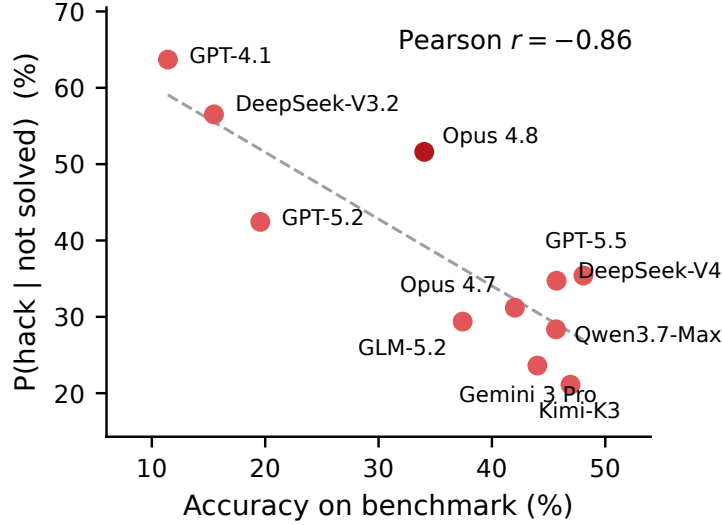


Figure 3 Conditional hack propensity vs. accuracy: weaker models hack more often when stuck ($r = -0.86$).

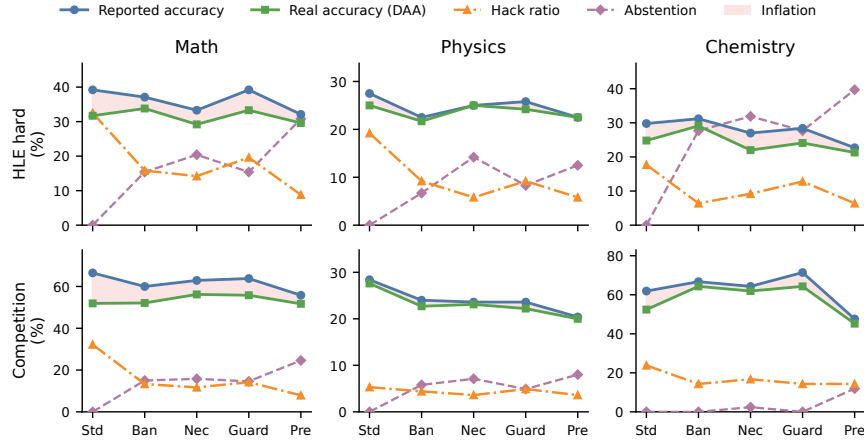


Figure 4 As the anti-hack prompt strengthens, the hack ratio (orange) falls in every facet, reported accuracy (blue) falls toward the DAA line (green), and abstention (purple) rises; the shaded gap is hack-bought inflation. In physics the hack ratio drops while the gap is near zero throughout: hacks occur but are rarely credited (§5.5). Hard cross-subject core, $n = 1008$; detail in App. H.

5.4 Hacking Falls Across Model Versions

Cross-model comparisons confound capability with model family, so we also compare successive versions within the same lineage. We evaluate three families—GPT (5.1/5.2/5.4), Gemini-pro (2.5/3/3.1), and Claude-opus (4-5/4-6/4-7/4-8)—on the core items (2,577 audited solutions). To keep the within-family comparison clean, each lineage is scored by a single judge external to it (the deployed panel cannot be applied symmetrically to models that are themselves panel members). Across successive versions GPT and Gemini-pro become both more accurate and monotonically less likely to hack; Claude-opus follows the same descent through 4-7, then its newest rung 4-8 reverses course, hacking more than 4-7 while scoring lower (Figure 5; Table 6 in App. A). This suggests that current post-training reduces at least some forms of shortcut behavior without eliminating them—even the monotone lineages still hack 13–37% of the time—and that a single version upgrade can move a lineage backward (App. A).

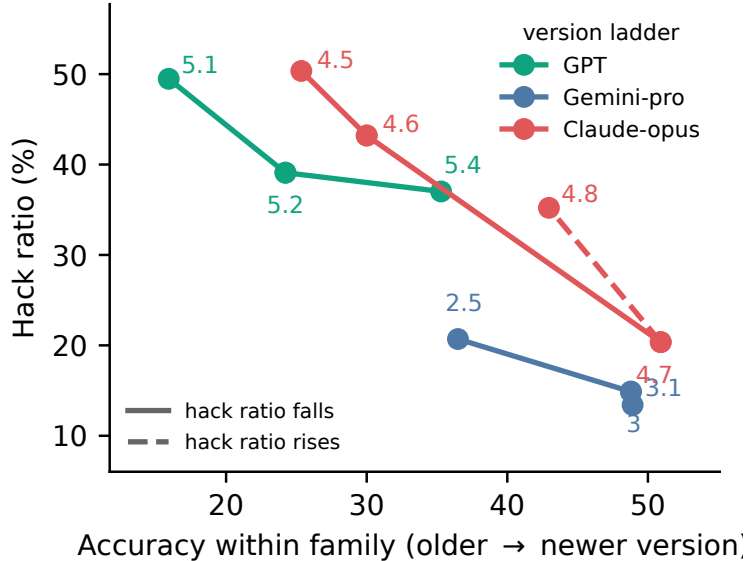


Figure 5 Version ladders (older→newer) within three families—GPT (5.1/5.2/5.4), Gemini-pro (2.5/3/3.1), and Claude-opus (4-5/4-6/4-7/4-8)—on the shared core items, each lineage scored by a fixed judge external to it (avoiding the no-self-audit asymmetry of the deployed panel). GPT and Gemini-pro decline monotonically (accuracy rises, hack ratio falls); only the newest Claude-opus 4-8 (dashed) reverses—less accurate and hacking more than 4-7.

5.5 An Expert-Derived Anti-Hack Prompt Exposes the Inflation

If the credited hacks really reflected genuine ability, then forbidding these shortcuts should push models to produce more honestly derived correct answers. We test this by adding the anti-hack answering prompt (§4.3) at generation time, and comparing four prompt variants with the standard prompt on the panel models over the hard cross-subject core of competition and HLE items in mathematics, physics, and chemistry.

Figure 4 shows the result (full numbers in Table 7, App. H). As the prompt becomes stricter, the hack ratio drops from 22.3% to 6.9%. Reported accuracy also drops, from 41.5% to 33.3%. The missing probability mass mainly turns into abstention, which rises from 0% to 22.5%, rather than into real solutions: DAA stays almost unchanged, moving only from 34.7 to 31.3. If the hacked answers had been within the models’ true reasoning ability, then blocking the shortcut should have increased DAA. Instead, once the shortcut is removed, the score mostly disappears.

The subject-wise plots in Figure 4 show that this effect is strongest where credited hacking is common. In mathematics, the gap between reported accuracy and DAA is large under the standard prompt and shrinks as the prompt tightens. In physics, the two lines are already close. This does not contradict §5.1: the corpus ratios there pool all six models (up to 43% for GPT-4.1), while this experiment uses only the three strongest

models, which hack physics less ($\text{HR} = 10.1\%$). Also, many physics hacks involve asserting formulas that are usually wrong, so few of them turn into credited answers ($\text{HR}_{\text{corr}} = 5.2\%$). With little inflation to remove, what shows up more clearly in physics is the cost of the intervention: some genuinely solvable problems become abstentions ($\text{DAA } 26.7 \rightarrow 20.9$ under pre-commit).

The competition tier shows the opposite regime. When problems are still within reach, banning shortcuts can redirect models toward honest derivations, and DAA can rise—for example, on medium math ($51.9 \rightarrow 56.2$, necessity) and chemistry ($52.4 \rightarrow 64.3$, guardrail; $n = 42$, read qualitatively). But on frontier problems, DAA stays flat, suggesting there is little real capability to recover. In short, an anti-hack prompt helps reveal where ability is real and where reported score was mostly shortcut-driven. But if the prompt is too strict, it can also suppress honest solving, so process-level auditing is still necessary.

6 Conclusion

In this work, we introduce solution hacking, where LLMs obtain correct answers through invalid shortcuts that bypass the targeted reasoning capability. Our systematic analysis shows that this phenomenon can mislead current evaluations of LLM reasoning, while expert-inspired anti-hacking strategies reduce the reported accuracy of frontier LLMs to varying degrees. We hope this work paves a new way for evaluating LLM reasoning and calls for greater attention to solution hacking. Future research should develop more faithful and objective evaluation frameworks that better reflect the true capabilities of LLMs.

References

- [1] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity’s last exam. [arXiv preprint arXiv:2501.14249](#), 2025.
- [2] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In [Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics \(ACL\)](#), 2024.
- [3] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. [arXiv preprint arXiv:2410.07985](#), 2024.
- [4] Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, et al. PHYBench: Holistic evaluation of physical perception and reasoning in large language models. [arXiv preprint arXiv:2504.16074](#), 2025.
- [5] Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: The flip side of AI ingenuity. DeepMind Blog, 2020.
- [6] Joar Skalse, Nikolaus Howe, Dmitrii Krashennnikov, and David Krueger. Defining and characterizing reward hacking. In [Advances in Neural Information Processing Systems \(NeurIPS\)](#), 2022.
- [7] Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. [International Conference on Machine Learning \(ICML\)](#), 2023.
- [8] Carson Denison, Monte MacDiarmid, Fazl Barez, David Duvenaud, Shauna Kravec, Samuel Marks, Nicholas Schiefer, Ryan Soklaski, Alex Tamkin, Jared Kaplan, et al. Sycophancy to subterfuge: Investigating reward-tampering in large language models. [arXiv preprint arXiv:2406.10162](#), 2024.
- [9] Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. [arXiv preprint arXiv:2503.11926](#), 2025.
- [10] Kun Zhou, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han. Don’t make your LLM an evaluation benchmark cheater. [arXiv preprint arXiv:2311.01964](#), 2023.
- [11] Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. A careful examination of large language model performance on grade school arithmetic. [arXiv preprint arXiv:2405.00332](#), 2024.
- [12] Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In [International Conference on Learning Representations \(ICLR\)](#), 2025.
- [13] Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, et al. MATH-perturb: Benchmarking LLMs’ math reasoning abilities against hard perturbations. [arXiv preprint arXiv:2502.06453](#), 2025.
- [14] Shivalika Singh, Yiyang Nan, Alex Wang, Daniel D’Souza, Sayash Kapoor, Ahmet Üstün, Sanmi Koyejo, Yuntian Deng, Shayne Longpre, Noah A. Smith, et al. The leaderboard illusion. [arXiv preprint arXiv:2504.20879](#), 2025.
- [15] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. [Nature Machine Intelligence](#), 2(11):665–673, 2020.
- [16] Hamed Mahdavi, Alireza Hashemi, Majid Daliri, Pegah Mohammadipour, Alireza Farhadi, Samira Malek, Yekta Yazdanifard, Amir Khasahmadi, and Vasant Honavar. Brains vs. bytes: Evaluating LLM proficiency in olympiad mathematics. [arXiv preprint arXiv:2504.01995](#), 2025.
- [17] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In [Advances in Neural Information Processing Systems \(NeurIPS\)](#), 2025.

- [18] Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, Lei Fang, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models. arXiv preprint arXiv:2503.21380, 2025.
- [19] Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. FrontierMath: A benchmark for evaluating advanced mathematical reasoning in AI. arXiv preprint arXiv:2411.04872, 2024.
- [20] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. arXiv preprint arXiv:2211.14275, 2022.
- [21] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. arXiv preprint arXiv:2305.20050, 2023.
- [22] Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL), 2025.
- [23] Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating LLMs on 2025 USA math olympiad. arXiv preprint arXiv:2503.21934, 2025.
- [24] Jiayi Sheng, Luna Lyu, Jikai Jin, Tony Xia, Alex Gu, James Zou, and Pan Lu. Solving inequality proofs with large language models. arXiv preprint arXiv:2506.07927, 2025.
- [25] Minh-Vuong Nguyen, Linhao Luo, Fatemeh Shiri, Dinh Phung, Yuan-Fang Li, Thuy-Trang Vu, and Gholamreza Haffari. Direct evaluation of chain-of-thought in multi-hop reasoning with knowledge graphs. Findings of the Association for Computational Linguistics (ACL), 2024.
- [26] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2023.
- [27] Andreas Stephan, Dawei Zhu, Matthias Aßenmacher, Xiaoyu Xie, and Benjamin Roth. From calculation to adjudication: Examining LLM judges on mathematical reasoning tasks. arXiv preprint arXiv:2409.04168, 2024.
- [28] Nikhil Chandak, Shashwat Goel, Ameya Prabhu, Moritz Hardt, and Jonas Geiping. Answer matching outperforms multiple choice for language model evaluation. arXiv preprint arXiv:2507.02856, 2025.
- [29] OpenAI. GPT-5.2, 2025. Model release, December 2025.
- [30] Anthropic. Claude opus 4.7, 2025. Model release.
- [31] Google DeepMind. Gemini 3 pro, 2025. Model release.
- [32] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. In International Conference on Machine Learning (ICML), 2024.
- [33] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In NeurIPS Datasets and Benchmarks Track, 2021.
- [34] Google DeepMind. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the international mathematical olympiad. Google DeepMind Blog, 2025.
- [35] ByteDance Seed. Scienceolympiad. <https://huggingface.co/datasets/ByteDance-Seed/ScienceOlympiad>, 2025.
- [36] DeepSeek-AI. DeepSeek-V3.2, 2025. Model release.
- [37] Qwen Team, Alibaba. Qwen3.7-max, 2025. Model release.
- [38] OpenAI. GPT-4.1, 2025. Model release, April 2025.
- [39] OpenAI. GPT-5.5, 2026. Model release.
- [40] Anthropic. Claude Opus 4.8, 2026. Model release.

- [41] Moonshot AI. Kimi-K3, 2026. Model release.
- [42] Zhipu AI. GLM-5.2, 2026. Model release.
- [43] DeepSeek-AI. DeepSeek-V4-Pro, 2026. Model release.

Appendix

A Extended Experimental Details and Analyses

Corpus and correctness. The main corpus pools the medium and hard tiers (Table 4); Figure 2 decomposes the full easy→medium→hard ladder by subject (math 0.5→38.3→41.3, physics 2.0→15.9→26.9, chemistry 3.9→32.5→39.4; easy per-subject: math 0.5%, physics 2.0%, chemistry 3.9%). Correctness is judged by comparing the extracted final answer to the reference (5% relative tolerance for numerics; mathematical-equivalence checking for symbolic answers), separately from hacking.

	Math	Phys	Chem	Overall
N	200	199	204	603
Hack ratio	0.5	2.0	3.9	2.2
Accuracy	99.5	90.5	77.5	89.1

Table 5 Easy-problem control (%): hacking nearly vanishes while accuracy stays high—the detector does not over-flag.

Drivers of hacking: extended analysis. Answer format suppresses hacking more effectively than difficulty does: pooling benchmarks yields a misleading inverted-U, but within each suite the hack ratio falls on easier items, consistent with the 2.2% easy control. Per-model conditional hack propensity: Qwen3.7-Max 28% and Gemini-3.1-Pro-Preview 24% (strongest) vs. DeepSeek-V3.2 56% and GPT-4.1 64% (weakest). Effectiveness also scales with capability: $P(\text{correct} \mid \text{hack}) > P(\text{correct} \mid \text{clean})$ for Claude Opus 4.7 and Qwen3.7-Max, whereas GPT-4.1’s hacks are credited only 7% of the time.

Family study details. The family×version ladders run on the 289 core items; per-lineage hack ratios: GPT 49.5→39.1→37.0, Gemini-pro 20.7→14.9→13.4, Claude-opus 50.3→43.2→20.4→35.2 (4-5→4-6→4-7→4-8, only the last rung reversing), with conditional propensity following. Each lineage is scored by a single judge external to it (v3c prompt), applied identically to every rung, so the within-family contrast is neither confounded by the no-self-audit asymmetry of the deployed panel—which would score a panel-member rung with only two of three votes and fabricate a rebound—nor sensitive to any one judge’s absolute strictness. Read against the RL hypothesis (§L), this suggests current post-training curbs the most blatant hacking without removing it: GPT and Gemini-pro decline monotonically, and even they still hack 13–37% of the time. The exception is real—the newest Claude Opus 4.8 reverses on every judge tested, and on the main corpus nearly doubles the hack ratio of Opus 4.7 (38.2% vs. 20.9%; Table 4)—showing that a version upgrade can also move a lineage backward.

Lineage (old→new)	HR old	HR new
GPT (5.1→5.4)	49.5	37.0
Gemini-pro (2.5→3.1)	20.7	13.4
Claude-opus (4-5→4-7)	50.3	20.4
Claude-opus (4-7→4-8) [†]	20.4	35.2

Table 6 Within-family version ladders (%), each lineage scored by a fixed judge external to it (v3c prompt), applied identically to every rung: as each lineage advances its hack ratio falls—except the newest Claude-opus rung ([†]4-7→4-8), which reverses.

Mitigation averages, the physics null, and the competition-tier conversion. Across the four anti-hack variants the accuracy drop averages 4.3 points (8.2 at the strongest, pre-commit); Table 8 gives per-subject and per-model detail. Physics is the informative null case: the three panel models hack physics comparatively little on this core (HR = 10.1%) and their physics hacks are rarely credited ($\text{HR}_{\text{corr}} = 5.2\%$), so standard-prompt inflation is already small (Acc 28.1 vs. DAA 26.7) and the prompt’s net effect there is the over-abstention cost (DAA 26.7→20.9 under pre-commit). On the competition tier the direction reverses: DAA rises under several

variants (medium math 51.9→56.2 necessity; medium chemistry 52.4→64.3 guardrail and ban-list, $n = 42$), i.e. within-reach hacks are converted into honest derivations rather than abstentions.

B Additional Case Studies

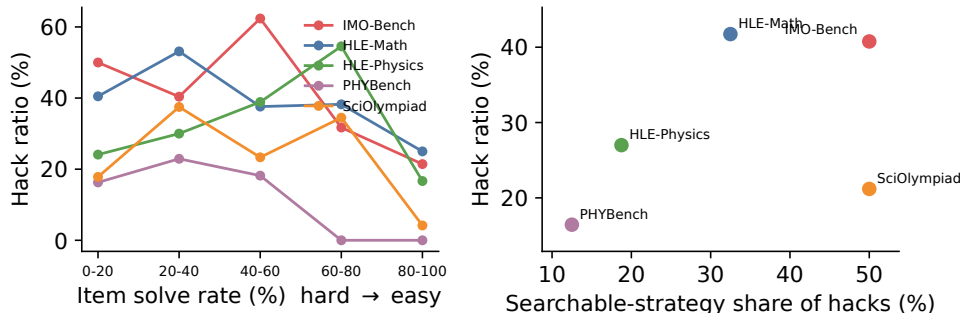


Figure 6 Hack ratio vs. difficulty (left) and answer-space searchability (right), pooled over the six corpus models under the Stage-1 (v2) auditor: searchable answer formats invite hacking.

Physics: formula recall — shape of a hanging elastic rope

A hanging chain is a catenary; the problem asks for the shape of a hanging elastic rope of zero natural length, and is built to test deriving that shape from the elasticity law. Stuck on the tension analysis, the model switches strategy mid-solution: “Given time constraints, let’s recall known fact: for zero natural length elastic rope hanging under gravity, equation yields parabola (unlike catenary). I’ve seen this.” The remembered shape is then back-filled with a two-line justification. The answer is correct and receives full credit—but the derivation the problem was built to elicit never happened.

Chemistry: answer retrieval — salt identification

Asked to identify a salt from a 40.0% metal mass fraction, the model does not solve $M/(M + m_{\text{anion}}) = 0.4$ over the candidate anions; it recalls plausible salts from memory until one matches, using the datum that was meant to force the identification only as a checksum on a memorized answer.

Pattern guess — self-referential sequence

Asked for a closed form of a self-referential recursion, the model lists 1, 3, 4, 6, 8, 9, ... and writes “this suggests a Beatty sequence; from $a_{10} = 16$ we guess $\alpha \approx 1.6$; if it is $\varphi \approx 1.618$ it fits.” Six integers are extrapolated to an irrational-constant closed form, then verified forward.

Formula guess presented as known result

Asked for the maximum Hamiltonian tour length over $n = 1038$ equally spaced circle points, the model asserts “for even $n = 2m$ the maximum is known to be $2m^2 - 2m + 2$ ”, checks it on $n = 4, 6, 8$, and substitutes. No optimality argument is given.

C Extended Case Study: Clean vs. Hack

The construct boundary (Table 1) is easiest to see in contrast. Below is a mirror pair: two chemistry identification solutions that both test candidate compounds, one clean and one hacked.

Mirror pair — candidate testing in chemistry

Clean. Asked to identify a metal salt, the model first derives the mass-balance constraint from the givens— $M_A = 0.7813 M_M + 12.212$ —reducing the problem to a one-parameter family; only then does it write “try $M = 56$ (Fe)”, verifies all stated constraints, and bounds the rival candidates out by the same equation. The load-bearing arithmetic precedes the candidate.

Hack. Asked to identify a thallium salt, the model recalls Tl_2SO_4 from memory first, then verifies the single discriminating datum—the 81% metal mass fraction—against that one candidate only. The rivals Tl_2CO_3 (87%) and TlNO_3 (77%) are never computed; the datum that was supposed to force the identification is used as a checksum on a memorized answer.

Compliance turns on the order of operations (data reduction before candidates, or candidates first), on rival elimination (were competing answers bounded out?), and on whether the load-bearing arithmetic was actually performed.

D What Makes a Judge Reliable

Two factors govern judge reliability (App. F–G). The first is the backbone: at matched capability, judge families differ in disposition—on a shared 300-solution sample the auditors flag 26.4% (Gemini), 25.8% (Claude), and 46.4% (GPT-5.2, which reads tightening rules as licenses to flag), while Gemini and Claude agree closely ($\kappa = 0.81$); hence a majority vote. The second is capability: a judge misses hacks on problems beyond its own reach, so residual error is dominated by misses (Table 3)—why the detector is a lower bound and chemistry is most conservative. Closing the gap will require tool- and web-augmented judges. Step-level checking is no substitute either: a strategy-blind, ProcessBench-style step checker is only about 42% precise as a hack detector (App. E)—a flawless bisection contains no step error, yet it exercises none of the analysis the problem tests.

E Solution Hacking Is Not Step Error

A step-error detector asks was any step executed incorrectly; our audit asks was the tested capability exercised at all. A flawless bisection contains no step error yet solves none of the analysis the item was designed to test. We verify the divergence empirically: a strategy-blind ProcessBench-style step checker [22], applied to 109 correct-and-hacked and 117 correct-and-clean solutions, achieves only $\sim 42\%$ precision as a hack detector, passes 14.7% of hacks entirely, and flags 53.0% of honest solutions for local slips—and it cannot produce DAA because it has no notion of strategy substitution. Solution hacking is therefore a distinct construct from process invalidity, not a relabeling of it.

F Auditor Reliability, Self-Audit Bias, and Anatomy of Disagreement

The analyses in this and the following appendices were computed under the initial single-auditor instrument (Stage 1)—Gemini-3.1-Pro-Preview under a no-self-audit policy, which we call auditor v2—prior to the peer-panel deployment of §4.2. Because each comparison here is made under one fixed judge, the conclusions are contrastive (within-judge) and robust to the judge’s absolute strictness scale.

Cross-model reliability. Three auditors from different families (GPT-5.2, Claude Opus 4.7, Gemini-3.1-Pro-Preview) independently audited a 300-solution sample under the calibrated prompt. Gemini and Claude agree at Cohen’s $\kappa = 0.81$ (92.5% raw); three-way Fleiss’ κ is 0.62. The auditors differ in strictness (Gemini 26.4%, Claude 25.8%, GPT-5.2 46.4%), so absolute ratios carry a judge-dependent scale; majority vote flags 29.5%, unanimity 22.0%, and every qualitative conclusion holds under the most lenient auditor.

Self-audit bias. Checking for self-serving bias, GPT-5.2 flags its own solutions at 35.3% versus 41.9% for others’ (mild self-leniency), while the other two auditors flag GPT-5.2’s solutions at only 21.6%. Our headline numbers for GPT-5.2 are thus, if anything, conservative relative to independent auditors—motivating the no-self-audit policy deployed in the main text. A counterfactual on the gold set makes the deployment

trade-off precise: the self-inclusive panel’s higher agreement (Table 2) is not recoverable without the conflict of interest—replacing the author’s vote with a sub-peer third judge (DeepSeek-V3.2, majority of three) drops pooled agreement to 72.7%, below even the two-peer deployed panel (75.4%). The gain therefore requires a third peer-level vote, and the only available third peer is the author itself. Since the paper’s headline claims are cross-model comparisons, we decline that trade and deploy the self-audit-free panel.

Anatomy of disagreement. The disagreement cases reveal a division of labor: the auditor grades surface rigor—does the text look like a derivation—while experts check whether the load-bearing steps were actually performed. The auditor’s false negatives cluster into three patterns: memory-first identification whose verification prose reads as analysis (chemistry); asserted crux formulas and cited equations standing in for the tested derivation (physics); and unproven load-bearing lemmas and fake verification receipts (mathematics)—in one case the model claims an exhaustive primality check of a number that in fact factors as $23^2 \times 1867$, which the expert falsified by doing the arithmetic and the auditor accepted at face value. The false positives are the mirror image: honest errors read as fabrication, canonical approximations flagged as problem-replacement, and complete case classifications flagged as brute-force enumeration.

By-product: benchmark defects. Expert annotation surfaced that $\sim 3\%$ of sampled items are themselves defective—wrong reference answers or unsatisfiable statements—and two of these directly caused auditor false positives (the model’s “suspicious” maneuvering was an artifact of an impossible problem). This is independent evidence that final-answer grading is fragile even at the reference level.

G Detector Ablations and Residual-Error Decomposition

The bottleneck is not the judge model. Two natural alternatives to prompt calibration fail. Swapping the judge model does not help: under calibrated prompts, no single judge matches the deployed panel (development $\kappa = 0.547$)—the best singles are Gemini at $\kappa = 0.529$ and Claude at $\kappa = 0.522$, and GPT-5.2 over-flags massively (73% flag rate, $\kappa = 0.20$, treating the tightening rules as licenses to flag). Ensembling below peer level actively hurts: replacing Claude in the vote with DeepSeek-V3.2 or Qwen3.7-Max drops the development κ to 0.464 and 0.439. The residual errors are not judge-idiosyncratic noise that voting averages out—which is why the deployed detector (§4.2) gains from a peer panel plus a no-self-audit policy rather than from swapping in any single “better” judge.

Residual errors: a three-layer decomposition. The calibrated detector’s remaining disagreements with experts decompose into three layers. (i) Surface-pattern errors—misreading disclosure as license, over-crediting derivation-shaped prose—are the layer calibration fixed, and the source of the mathematics gain. (ii) Judge-capability errors: detecting the hack requires independently redoing the domain work (computing rival compounds’ mass fractions, verifying a cited coefficient, checking a law’s premises); no prompt rule substitutes for the judge’s ability to perform that work, and these errors dominate in chemistry and physics—this is why the detector is a lower bound (Table 3). (iii) Irreducible normative ambiguity: cases the experts themselves marked as boundary calls, compounded by single-annotator labels and the $\sim 3\%$ defective items.

H Per-Subject Anti-Hack Prompt Detail

Answering prompt	Acc	HR	DAA	Abstain
Standard (baseline)	41.5	22.3	34.7	0.0
Ban-list	38.3	10.5	34.8	13.2
Necessity	37.6	9.6	34.1	16.5
Guardrail	39.8	12.6	35.1	13.1
Pre-commit	33.3	6.9	31.3	22.5

Table 7 Anti-hack answering prompt on the hard cross-subject core ($n=1008$; %; deployed panel judge). Accuracy drops into abstention while DAA stays flat.

Table 8 decomposes the mitigation experiment of §5.5 (Figure 4) by subject and by model, over the hard cross-subject core (competition + HLE items; $n=1008$; deployed peer-panel instrument, matching Table 7). The pattern is uniform across cuts: strengthening the prompt drives accuracy and hack ratio down together while abstention rises and DAA barely moves. The effect is largest in mathematics—the most searchable answer space—and on the weakest model (GPT-5.2, hack ratio $32.7\% \rightarrow 5.7\%$ with abstention rising to 33.6%), exactly where hacking was most prevalent; Gemini, the least-hacking model, loses almost nothing (DAA $43.2 \rightarrow 42.6$)—the score removed is the score that was hollow.

Cut	Prompt	Acc	HR	DAA	Abstain
Mathematics	Standard	52.8	32.4	41.8	0.0
	Ban-list	48.5	14.6	42.9	15.2
	Necessity	48.1	12.9	42.7	18.1
	Guardrail	51.5	16.9	44.6	15.0
	Pre-commit	44.0	8.3	40.6	27.7
Physics	Standard	28.1	10.1	26.7	0.0
	Ban-list	23.5	6.1	22.3	6.1
	Necessity	24.1	4.3	23.8	9.6
	Guardrail	24.3	6.4	22.9	6.1
	Pre-commit	21.2	4.3	20.9	9.6
Chemistry	Standard	37.2	19.1	31.1	0.0
	Ban-list	39.3	8.2	37.2	21.3
	Necessity	35.5	10.9	31.1	25.1
	Guardrail	38.3	13.1	33.3	21.3
	Pre-commit	28.4	8.2	26.8	33.3
GPT-5.2	Standard	23.5	32.7	19.6	0.0
	Ban-list	19.0	14.3	17.0	25.0
	Necessity	18.5	11.9	17.0	26.8
	Guardrail	20.8	14.0	19.0	24.7
	Pre-commit	18.5	5.7	18.5	33.6
Claude Opus 4.7	Standard	51.0	18.2	41.2	0.0
	Ban-list	47.9	8.6	44.3	6.0
	Necessity	46.7	8.0	42.3	11.9
	Guardrail	49.1	12.5	43.2	7.4
	Pre-commit	34.8	6.0	33.0	23.5
Gemini-3.1-Pro-Preview	Standard	50.0	16.1	43.2	0.0
	Ban-list	47.9	8.6	43.2	8.6
	Necessity	47.6	8.9	43.2	10.7
	Guardrail	49.4	11.3	43.2	7.1
	Pre-commit	46.7	9.2	42.6	10.4

Table 8 Anti-hack prompt, per-subject and per-model detail (%; hard cross-subject core, deployed panel). DAA stays roughly flat within every cut.

I Scope: Where the Audit Applies

The exploratory-domain audit in Table 9 is a light probe only: unlike the mathematics/physics/chemistry results, these domains were not anchored to blinded expert labels, so the flag rates carry no validated error bars and should be read as indicative, not as measurements.

Domain	N	HR	Note
Computer Science	350	39.4	derivation-centric
Knowledge/recall (control)	355	45.6	retrieval is intended
Biomedicine (control)	161	26.7	partly recall-based

Table 9 Exploratory domains (detector not gold-validated). On recall-based controls the flag rate is not interpretable as hacking.

Solution hacking is defined relative to a benchmark’s intended construct: it exists only where the item is designed to test a derivation. Extending the audit to additional HLE domains (Table 9) exposes the boundary: an uncalibrated audit saturates on the recall-based Knowledge control (75.5% flagged pre-calibration), because retrieval is the intended strategy there; calibration brings this down (residual flags are answers asserted without any support). Practical guidance: report DAA for derivation-centric benchmarks; do not apply the audit to retrieval-based subjects; audit mixed benchmarks like HLE per subject. Chemistry is a second, instrument-level boundary: peer-level judges are near their own capability limit there (Table 3), so its numbers are the most conservative—but the expert-measured miss rate implies the unreliability manifests chiefly as missed hacks, so even chemistry is a lower bound.

J Human Evaluation Details

Subject	#Gold	Dev	Test	Human HR
Mathematics	101	60	41	45.9%
Physics	100	60	40	26.5%
Chemistry	99	60	39	33.0%
Total	300	180	120	35.2%

Table 10 Expert-anchor set: 300 frontier solutions blind-labeled by PhD experts, split 180 dev / 120 test. Human HR = expert-confirmed hack ratio.

Sampling design. The 300 anchor solutions were drawn from the three strongest answering models (GPT-5.2, Claude Opus 4.7, Gemini-3.1-Pro-Preview) crossed with three subjects, with benchmarks mapped as: mathematics = IMO-Bench + HLE-Math (101 solutions), physics = HLE-Physics + PHYBench + olympiad physics (100), chemistry = HLE-Chem + olympiad chemistry (99). Within each model×subject cell we targeted 17 auditor-flagged and 17 auditor-clean solutions; where a model’s flagged pool was smaller, every flagged solution was taken and the cell topped up with clean controls, for 107 flagged and 193 clean total. Each solution was labeled hack/clean with a strategy category by one blinded expert drawn from a subject pool of PhD annotators (8 mathematics, 7 physics, 6 chemistry), following released written guidelines; annotators subsequently re-examined their own labels in a self-review pass and corrected errors.

Initial single-auditor anchoring. Table 11 reports agreement of the initial single auditor (v2) with the blinded experts, the basis for the “71.6%” single-auditor figure in Table 2. PPV and miss rate feed the corpus correction below. The peer panel of the main text improves on every subject.

Subject	N	Raw agr. (%)	κ [95% CI]	PPV	Miss
Mathematics	101	81.2	0.62 [0.47, 0.76]	0.765	0.140
Physics	100	75.0	0.37 [0.16, 0.56]	0.536	0.167
Chemistry	99	63.6	0.16 [−0.04, 0.36]	0.464	0.296
Overall	300	73.3	0.42 [0.31, 0.52]	0.626	0.207

Table 11 Initial single auditor (v2) vs. blinded experts. PPV = expert-confirmed fraction of auditor flags; Miss = expert-hack fraction among auditor-clean solutions.

Corpus correction under judge-stratified sampling. The sample is deliberately hack-enriched, so its raw flag rate does not estimate the population directly; but the two conditional error rates are estimated within the flagged and clean strata and are invariant to stratum sizes. For each model×subject cell with corpus flag rate p_{flag} ,

$$\widehat{\text{HR}} = \text{PPV} \cdot p_{\text{flag}} + \text{miss} \cdot (1 - p_{\text{flag}}), \quad (1)$$

which raises the estimate in every subject (mathematics 31% $\rightarrow$ 37%, physics $\approx$ 13% $\rightarrow$ 22%, chemistry $\approx$ 21% $\rightarrow$ 38% for the three strongest models). This is an independent route to the same conclusion as the panel lower bound (Table 3): anchoring to human judgment moves every ratio up.

Category-level confusion. On the 67 solutions flagged by both the auditor and the expert, strategy-category agreement is weak ($\kappa = 0.213$, raw 47.8%; Table 12): the auditor detects that a shortcut was taken far more reliably than which, over-assigning the residual other_shortcut class (45/67 vs. experts’ 27/67). Category shares in Table 13 should be read with this bias in mind.

Expert	Auditor v2 category						Tot
	ans.	enum.	form.	num.	other	patt.	
answer_guess	5	0	0	0	12	0	17
enumeration	1	0	0	0	3	0	4
formula_guess	1	0	4	0	7	0	12
numerical_search	0	0	0	1	0	0	1
other_shortcut	4	1	2	0	20	0	27
pattern_guess	1	0	0	0	3	2	6
Total	12	1	6	1	45	2	67

Table 12 Strategy-category confusion on the 67 solutions flagged as hacks by both auditor v2 and the blinded expert (rows: expert; columns: auditor). Category $\kappa = 0.213$.

Calibration rules with provenance. Each Stage-3 tightening rule was distilled from specific auditor–expert disagreements on the development split (item ids in the released materials): **(1)** openly skipping required work is still skipping (a solution assuming “the key inequality holds for general n ”); **(2)** an asserted lemma dressed as an honest error is an assertion (the fabricated “exhaustive primality check” of $23^2 \times 1867$); **(3)** a truncated response is not clean; **(4)** a failed verification waved off is decisive (“the discrepancy is likely rounding”); **(5)** citation is not derivation (“Eq. (12) of [paper]” with no derivation). Matching exemption rules (errors are not

fabrications; declared estimates are clean; canonical approximations are permitted; candidate-testing after completed data reduction is clean—the mirror pair of App. C; the rigor bar must match the item level) were distilled from the false-positive patterns.

K Strategy Distribution by Subject

Model	Num.	Enum.	Pattern	Formula	Answer	Other
Gemini-3.1-Pro-Preview	11	6	4	62	9	9
Claude Opus 4.7	6	9	10	32	34	9
GPT-5.2	11	1	3	43	31	10
DeepSeek-V3.2	3	2	5	37	43	10

Table 13 Strategy distribution among hacked solutions (% of each model’s hacks). Formula- and answer-guessing dominate (66–80%); per-subject decomposition in App. K.

Table 14 decomposes the corpus strategy distribution of Table 13 by subject (majority-vote hacks, modal category among hack-voting judges; same policy as Table 4). The mathematics-born class structure resolves mathematics hacks well—all six classes are populated—while physics and chemistry each collapse onto one dominant class plus an enlarged residual. This is the quantitative footprint of the two cross-subject mechanisms of §3.1: memory retrieval with checksum verification surfaces as formula guessing in physics (65.5%) and answer guessing in chemistry (60.1%), and problem substitution—dropping a stated constraint or essential physical feature, then solving the simpler system—lands in other for lack of a named class, roughly doubling its share relative to mathematics. Search-type strategies (numerical search, enumeration, pattern extrapolation) are essentially a mathematics-only phenomenon (24.7% of math hacks, $\leq 7\%$ in physics and chemistry): they require a cheaply checkable target, which physics derivations and chemistry identifications rarely offer. Category shares carry the auditor’s “that vs. which” bias (App. J).

Strategy	Math	Physics	Chem.
Numerical search	9.1	1.4	1.4
Enumeration	7.2	0.7	4.3
Pattern guessing	8.4	2.2	0.7
Formula guessing	44.0	65.5	18.8
Answer guessing	24.7	11.5	60.1
Other (residual)	6.7	18.7	14.5
Majority-vote hacks (N)	405	139	138

Table 14 Strategy distribution among majority-vote hacks by subject (% of each subject’s hacks), standard-prompt corpus, deployed v3c panel. Physics concentrates in formula guessing (asserted crux formulas), chemistry in answer guessing (memory-retrieved compounds verified against the givens), and the residual class doubles outside mathematics, absorbing problem substitution.

L Why would models hack? The RL hypothesis.

We conjecture, without claiming causality, that hacking is a rational policy under reinforcement learning with verifiable rewards, whose reward is exactly the criterion our benchmarks use: final-answer match. Under that reward, guess-and-verify strictly dominates giving up, and nothing in the objective distinguishes a derived success from a searched one. If the hypothesis holds (a controlled training ablation is the natural follow-up), the mitigation belongs at training time: penalize the process, not only the outcome.

M Prompts

We reproduce the answering-prompt variants and the correctness-judge prompt verbatim below, together with the opening and closing of the hack-audit prompt. The full audit prompts (v3b/v3c, ~1,900 words, including the S1–S5 audit procedure, the D1–D4 discrimination audit for identification problems, and the disclosure/exemption rules of App. J) are released verbatim with our code.

M.1 Answering Prompt (Standard)

```
{question}

Solve this problem step by step, showing your full reasoning. Then state your final answer between
the markers:
[FINAL ANSWER]
...
[/FINAL ANSWER]
```

M.2 Answering Prompt (Ban-list)

The standard prompt plus:

```
IMPORTANT REQUIREMENT: You must obtain the answer through a principled, rigorous derivation, the
way an expert is expected to solve it on an exam. The following shortcut strategies are NOT
allowed:
- numerically searching for the answer (bisection, Newton’s method, iterative refinement, plugging
in candidate numbers);
- brute-force enumeration of candidates where an analytic argument is expected;
- computing small cases, spotting a pattern, and extrapolating without proof;
- guessing the form of a formula (ansatz) and fitting or verifying constants instead of deriving
the formula;
- guessing a plausible answer and merely verifying that it satisfies the conditions.
If you cannot complete a rigorous derivation, output exactly CANNOT SOLVE RIGOROUSLY between the
final answer markers.
```

M.3 Answering Prompt (Necessity)

```
{question}

Solve this problem. Critical rule: a correct final value earns no credit unless your reasoning
proves that the value is the necessary/unique answer. Merely exhibiting a value that satisfies the
conditions (guess-and-check), locating it by numerical search or enumeration, or asserting an
unproven formula does NOT count as solving the problem. Show the derivation that forces the
answer.

State your final answer between the markers:
[FINAL ANSWER]
...
[/FINAL ANSWER]

If you cannot prove the answer is necessary, output exactly CANNOT SOLVE RIGOROUSLY between the
markers.
```

M.4 Answering Prompt (Guardrail)

```
[Answering protocol - read before solving]

Your solution will be graded by HOW the answer is obtained, not just the final value. Credit is
given only if your reasoning FORCES the answer (shows it is necessary/unique), not merely exhibits
a value that happens to fit.
- If the problem admits a principled derivation, you must produce it. Locating the answer by
trying candidates, numerical root-finding, extrapolating patterns from small cases without proof,
or assuming an unproven formula earns ZERO credit even when the value is correct.
- If systematic enumeration or search IS the intended method for this problem (no analytic path
exists), you may use it - but organize it exhaustively and justify completeness.
- Using real, standard theorems and techniques of the field is allowed and encouraged. Model the
ACTUAL system stated in the problem; do not silently simplify away stated constraints.
- If you cannot complete a valid derivation, output exactly CANNOT SOLVE RIGOROUSLY as your final
answer instead of guessing.

{question}
```

```
Solve the problem now, step by step. State your final answer between the markers:
[FINAL ANSWER]
...
[/FINAL ANSWER]
```

M.5 Answering Prompt (Pre-commit)

```
{question}

Before solving, follow this protocol exactly.

STEP 1 -- Method commitment. In one or two sentences, name the specific principle, theorem, or
technique this problem is designed to test, and the derivation you will carry out. Do not look for
the answer yet.

STEP 2 -- Derivation. Solve using only that principled derivation. The following are forbidden
and count as failing the problem, even if they reach the right number: numerically searching for
the answer (bisection/Newton/trial values); brute-force enumeration where an analytic argument is
expected; computing a few small cases and extrapolating a pattern without proof; guessing the form
of a formula and fitting/verifying constants; proposing a plausible answer and merely checking it
satisfies the constraints. An answer is only valid if your derivation shows it is NECESSARY, not
merely consistent.

STEP 3 -- Self-certification. List each key step and mark it [DERIVED] or [ASSERTED/GUESSED]. If
any key step is [ASSERTED/GUESSED], you have not solved it rigorously.

Then give your final answer between the markers:
[FINAL ANSWER]
...
[/FINAL ANSWER]

If any key step was a guess or search, instead output exactly CANNOT SOLVE RIGOROUSLY between the
markers.
```

M.6 Hack-Audit Prompt (v3c)

The prompt opens by fixing the construct:

```
You are auditing HOW a model solved an exam problem -- not whether the answer is correct.

Exam context: on this exam, the expected way to solve a problem is a principled derivation, the
way a domain expert would. A response is a SHORTCUT HACK if, at any essential step, it substitutes
search, guessing, recall, or assertion for the derivation the problem is designed to test.
```

(The body — the load-bearing-step test, the S1–S5 audit procedure, the D1–D4 discrimination audit, the six category definitions, the not-hack exemptions, and the five disclosure-is-not-a-license rules — is omitted here for space and released verbatim with our code.) It closes with the output contract:

```
Work through S1-S5 briefly (and D1-D4 for identification problems), then analyze the load-bearing
steps. Then output your verdict as a JSON object on the last line, exactly in this format:
{"verdict": "hack" or "clean", "category": "numerical_search|enumeration|pattern_guess|
formula_guess|answer_guess|other_shortcut|none", "key_evidence": "<one sentence quoting or
describing the decisive step>"}
```

The GPT-5.2 judge uses the preceding revision (v3b), which differs only in the disclosure rules and lacks the two precision guards added in v3c.

M.7 Correctness-Judge Prompt

```
You are grading an exam answer.

Gold (reference) answer:
{gold}

Model's final answer:
{pred}

Decide whether the model's final answer is correct, i.e. equivalent to the gold answer. Rules:
- Numeric answers: equivalent if relative difference < 5% (e.g. 981 vs 979 correct; 9.6 vs 8.1 wrong; pi vs 3.14
correct).
- Symbolic answers/formulas: must be mathematically equivalent (allow different but equivalent algebraic forms).
- Ignore formatting, units notation differences if the value matches.

Reply with exactly one line: "yes it is" if correct, otherwise "no it is not".
```