Seirênes: Adversarial Self-Play with Evolving Distractions for LLM Reasoning

Chi Zhang[1], Haibo Qiu[2†], Qiming Zhang[3], Yufei Xu[1], Xinbo Gao[4], Jing Zhang[1✉]
[1]School of Computer Science, Wuhan University
[2]Meituan Inc [3]Independent Researchers [4]Xidian University
Project: `https://github.com/MiliLab/Seirenes`

Abstract

We present **Seirênes**, a self-play RL framework that transforms contextual interference from a failure mode of LLM reasoning into an internal training signal for co-evolving more resilient reasoners. While RL with verifiable rewards has significantly advanced reasoning capabilities, models can still exhibit fragility when encountering non-idealized contexts: scenarios characterized by superfluous information, tangential instructions, or incidental correlations that differ from the clean distributions typical of standard benchmarks. Seirênes harnesses this vulnerability through a parameter-shared and adversarial self-play loop. Within this framework, a single model is trained to both *construct plausible yet distracting contexts that expose its own reasoning blind spots*, and *solve problems by discerning the essential task from these perturbations to recover the core underlying logic*. By pitting these competing objectives against each other, Seirênes compels the model to move beyond superficial pattern matching and anchors its capabilities in robust underlying reasoning. This continuous interaction sustains an informative co-evolutionary curriculum as the model improves. Across seven mathematical reasoning benchmarks and model scales from 4B to 30B, Seirênes achieves average gains of +10.2, +9.1, and +7.2 points. Besides, distracting contexts produced by the 4B Seirênes model reduce the accuracy of top-tier closed-source models (GPT and Gemini) by roughly 4–5 points, revealing Seirênes' general ability to uncover reasoning models' blind spots.

1 Introduction

"You will come to the Seirênes first of all; they bewitch any mortal who approaches them."
— Homer, *Odyssey* XII, trans. Shewring [18]

Reinforcement learning with verifiable rewards (RLVR) has become the central driver of progress in LLM reasoning [27]. Propelled by a continuous streak of breakthroughs across complex reasoning domains, models have achieved competitive results in Olympiad-level mathematics and programming contests that once required years of human expertise [13, 51, 23, 19, 39, 2].

This rapid progress has renewed interest in self-play training for reasoning models. By coupling complementary roles within a training loop, such systems can dynamically generate curricula and supervision signals, echoing the historic success of AlphaGo [54]. Recent work such as R-Zero [20], Absolute Zero [78], and SPICE [33] demonstrates the power of this paradigm. Existing methods typically instantiate this game at the level of task generation or solution verification: one role proposes problems, programs, or supervision, while another learns to solve them.

[†]Project Leader. [✉]Corresponding Author: jingzhang.cv@gmail.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We explore a complementary form of self-play that leaves the task itself fixed but evolves the context surrounding it. This direction is motivated by prior evidence that strong reasoners remain brittle under seemingly minor contextual changes: MATH-P-Hard reduces GPT-4o's accuracy by 27.60 percentage points [21], biasing context can silently redirect chain-of-thought trajectories [55], and 80% retrieval noise lowers ChatGPT's English accuracy from 96.33% to 76.00% [6]. Taken together, these observations suggest that contextual interference can serve as more than an evaluation-time stress test. By exposing reasoning shortcuts that clean tasks may leave hidden, distracting contexts allow the curriculum to shift from *which* problems a model can solve to *how* it reasons through a problem. If a model can learn to generate such contexts against itself, those contexts can become an online training signal for improving the standalone reasoner.

We propose **Seirênes**, a self-play RL framework that turns contextual interference from a failure mode of LLM reasoning into an internal training signal driven by adversarial co-evolution. The name evokes the Seirênes of Greek mythology, whose alluring songs drew sailors away from their course; in our setting, the lure takes the form of plausible but misleading contextual hints. A single shared-parameter policy plays both an ***Adversary***, which generates plausible but misleading *adversarial hints*, and a ***Reasoner***, which must solve the original problem under the perturbed context. The Adversary is rewarded for derailing the Reasoner's solution trajectory, while the Reasoner is trained to anchor its deduction and recover the core problem logic under such interference. This adaptive training curriculum encourages Seirênes to continuously expose and solve new reasoning failures, thereby avoiding fragile reasoning shortcuts and improving robustness. Crucially, Seirênes modifies only the input context: the underlying task, verifier, and test-time interface remain unchanged, so the resulting model is optimized as a stronger standalone reasoner rather than as a special-purpose robustness module.

In summary, our main contributions are as follows:

- We introduce **Seirênes**, a self-play RL framework in which a single model co-evolves as both an *Adversary* that generates plausible but misleading adversarial hints and a *Reasoner* that learns to recover correct reasoning paths under such interference. This interaction turns contextual brittleness into an adaptive training curriculum.
- We develop a parameter-shared design that makes this adversarial loop trainable end to end, combining role-aware rewards, paired clean and hint-conditioned rollouts, role-specific update streams, and an efficient orchestration strategy that keeps cost tractable at scale.
- Extensive experiments show that Seirênes delivers consistent gains over both standard RL and cooperative-hint baselines on diverse mathematical reasoning benchmarks, generalizing to broader contextual perturbations. Controlled analyses attribute these gains to the dynamic adversarial loop, ruling out extra compute or static training curricula.

2 Related work

RLVR for Reasoning. RLVR is the central post-training paradigm for reasoning LLMs [27, 31]. While early studies demonstrate the efficacy of verifiable training signals [31, 58], DeepSeek-Math [51] marks a major algorithmic milestone by introducing GRPO, which is now foundational for open-weight RLVR research. OpenAI o1 [23] and DeepSeek-R1 [13] catalyze the current wave of reasoning models by demonstrating the massive potential of test-time scaling and long-horizon RL. DeepSeek-R1-Zero, in particular, advances the field by demonstrating that pure RL can directly elicit strong reasoning capabilities from base models without supervised fine-tuning [13]. A subsequent wave of research focuses on refining these pipelines, tackling core optimization challenges like training stability, value estimation, and reproducibility [69, 79, 19, 72, 38, 37, 14]. As baseline reasoning capabilities rapidly expand, a long-standing question gains immediate relevance: can models move beyond being optimized by static pipelines, and actively participate in generating the challenges and signals that drive their own continuous improvement?

Self-Evolving Reasoning. Recent literature explores this possibility by enabling models to synthesize their own curricula via autonomous self-play [20, 78, 33, 71, 62]. For instance, R-Zero investigates ungrounded challenger-solver co-evolution starting from zero data [20], Absolute Zero anchors self-play in executable environments with verifiable feedback [78], and SPICE introduces corpus-grounded self-play to sustain improvements through document-level information asymme-

try [33]. Subsequent extensions incorporate minimal human anchors (e.g., R-Few [71]) or tool integration (e.g., Agent0 [62]) to advance this paradigm. Similar concepts also appear in multimodal domains [35, 16]. Our work distinguishes itself by shifting the focus of co-evolution from task generation to the underlying reasoning process itself, leveraging the adversarial context surrounding a task as the primary catalyst for continued improvement.

Hint-Augmented RL and Intervention-Based Guidance. A closely related line of work augments RL training with hints, scaffolds, interventions, or other auxiliary guidance [63, 32, 76, 77, 68, 65]. HiLL and SAGE use adaptive helpful hints to densify reward or reshape the rollout distribution [63, 32]. StepHint derives multi-level stepwise hints from stronger reasoning traces [76]. Neighboring methods like Scaf-GRPO, InT and LUFFY further improve training through localized interventions or guidance from stronger external traces [77, 68, 65]. In stark contrast to these methods, our framework is fundamentally *adversarial* and *co-evolutionary*. A shared-parameter model competes against itself—simultaneously playing the *Adversary* to synthesize adversarial hints, and the *Reasoner* to overcome them. This attacker–target structure also connects to classical adversarial training [41, 24] and LLM red-teaming [48, 11, 80]. These areas primarily target robustness evaluation, attack defense, or safety tuning; we discuss the connection further in Appendix A.

3 Preliminaries

We build on Group Relative Policy Optimization (GRPO) [51], a critic-free policy gradient algorithm that replaces the learned value baseline with group-relative reward normalization. Given a question q, the behavior policy $\pi_{\theta_{\text{old}}}$ draws G rollouts:

$$\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q), \qquad R_i = R(q, o_i) \in \{0, 1\}, \tag{1}$$

where $R(q, o_i)$ is a binary verifiable reward (correct / incorrect). The advantage of each response is estimated by standardizing rewards within the group:

$$\hat{A}_i = \frac{R_i - \frac{1}{G}\sum_{j=1}^{G} R_j}{\text{std}\big(\{R_j\}_{j=1}^{G}\big) + \epsilon}. \tag{2}$$

All tokens within response o_i share the same outcome-level advantage $\hat{A}_{i,t} = \hat{A}_i$ [51, 69]. The policy is then updated by maximizing the clipped surrogate objective with a KL-divergence penalty:

$$J_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big(r_{i,t}(\theta)\,\hat{A}_i,\ \text{clip}(r_{i,t}(\theta), 1{-}\varepsilon, 1{+}\varepsilon)\hat{A}_i\Big) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right], \tag{3}$$

where the per-token importance ratio is $r_{i,t}(\theta) = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) \,/\, \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$, ε is the clipping threshold, and β controls the strength of the KL penalty toward the reference policy π_{ref}. When applying GRPO to a finite rollout group below, we write $J_{\text{GRPO}}(\theta; \mathcal{B})$ for the empirical version of Eq. 3 evaluated on batch $\mathcal{B}$.

4 Method

4.1 Role-conditioned rollout construction

To instantiate Seirênes, a single policy with parameters θ is prompted into two role-conditioned distributions: the Reasoner π_θ^{R} and the Adversary π_θ^{Adv}. For each training question q, Seirênes constructs a paired rollout bundle that compares the Reasoner's behavior on the original problem with and without an adversarially generated hint, as illustrated in Fig. 1. This process unfolds in three stages: obtaining a clean reference behavior (R1), synthesizing adversarial hints (R2), and evaluating the Reasoner under each of these interferences (R3). Together, this structured sampling yields a comprehensive trajectory bundle that isolates the causal impact of the adversarial context.

R1: Clean rollout. The policy first acts as the Reasoner on the original question q. It samples G_1 clean answer trajectories and computes their corresponding verifier rewards:

$$\{o_i^{(1)}\}_{i=1}^{G_1} \sim \pi_{\theta_{\text{old}}}^{\text{R}}(\cdot \mid q), \qquad R_i^{(1)} = R(q, o_i^{(1)}). \tag{4}$$

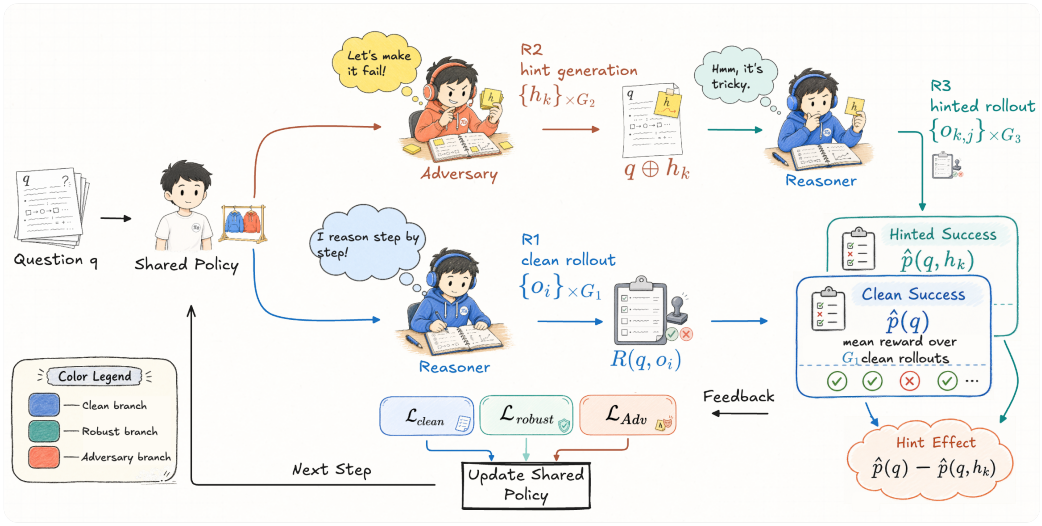

Figure 1: Seirênes: A single-policy internal arms race.

The empirical clean success rate is defined as:

$$\hat{p}(q) = \frac{1}{G_1} \sum_{i=1}^{G_1} R(q, o_i^{(1)}). \tag{5}$$

This establishes the Reasoner's unperturbed baseline capability. The estimate $\hat{p}(q)$ serves as an online difficulty signal for the current Reasoner: low values indicate that the model has not yet reliably solved the clean problem, whereas high values make subsequent failures under adversarial context more attributable to the hint rather than to baseline task difficulty.

R2: Adversarial hint generation. Operating as the Adversary, the shared policy samples G_2 hint candidates for the same question:

$$\{h_k\}_{k=1}^{G_2} \sim \pi_{\theta_{\text{old}}}^{\text{Adv}}(\cdot \mid q). \tag{6}$$

The Adversary is specifically prompted to generate context that is natural and locally plausible, deliberately steering the Reasoner toward an incorrect analytical path. Details of the exact role prompt are deferred to Appendix K.

R3: Hint-conditioned rollout. For each adversarial hint h_k, we construct an augmented prompt $q \oplus h_k$. The Reasoner role then answers this context, sampling G_3 trajectories per hint, and we obtain their rewards using the same ground-truth answer, as the hint introduces interference without altering the underlying problem:

$$\{o_{k,j}^{(3)}\}_{j=1}^{G_3} \sim \pi_{\theta_{\text{old}}}^{\text{R}}(\cdot \mid q \oplus h_k), \qquad R_{k,j}^{(3)} = R(q, o_{k,j}^{(3)}), \tag{7}$$

To quantify the Reasoner's robustness against this specific interference, we aggregate these rewards into an empirical hint-conditioned success rate:

$$\hat{p}(q, h_k) = \frac{1}{G_3} \sum_{j=1}^{G_3} R(q, o_{k,j}^{(3)}). \tag{8}$$

Together with the clean estimate in Eq. 5, this quantity establishes the exact performance drop needed to compute the adversarial reward.

4.2 Branch-specific training signals and update rules

The rollout bundle provides three trajectory types with distinct credit semantics: clean Reasoner answers, Adversary hints, and hinted Reasoner answers. We now define the branch-specific training signals and update rules associated with each stream.

Clean reasoning branch. The clean branch applies GRPO to the R1 answer trajectories with verifier rewards $R_i^{(1)}$:

$$\mathcal{L}_{\text{clean}} = -J_{\text{GRPO}}\Big(\theta; \{(q, o_i^{(1)}, R_i^{(1)})\}_{i=1}^{G_1}\Big). \tag{9}$$

By preserving ordinary RLVR training on the original task, Seirênes keeps the Reasoner anchored to its core problem-solving capabilities. This ensures the adversarial-context branches provide auxiliary robustness pressure without distorting the underlying task distribution.

Adversary branch. The Adversary is trained on the hint tokens generated in R2. We measure hint effectiveness by the performance drop from clean to hint-conditioned success. At the population level, for the behavior policy that generates the paired bundle, we define

$$p(q) = \mathbb{E}_{o\sim\pi^{\text{R}}_{\theta_{\text{old}}}(\cdot|q)}[R(q,o)]\,, \qquad p(q,h_k) = \mathbb{E}_{o\sim\pi^{\text{R}}_{\theta_{\text{old}}}(\cdot|q\oplus h_k)}[R(q,o)]\,. \tag{10}$$

Their difference defines the hint-effectiveness gap: $\Delta(q,h_k) = p(q) - p(q,h_k)$. In practice, Seirênes estimates this quantity empirically from the paired rollout bundle:

$$\hat{R}_{\text{Adv}}(q,h_k) = \hat{\Delta}(q,h_k) = \hat{p}(q) - \hat{p}(q,h_k). \tag{11}$$

Positive values indicate that the hint degrades the current Reasoner relative to its clean behavior. Beyond identifying degradation, this empirical gap provides a naturally bounded and asymmetric credit signal. Since $\hat{p}(q,h_k) \in [0,1]$, the reward lies in $[-(1-\hat{p}(q)), \hat{p}(q)]$. This structural property elegantly prevents common failure modes in adversarial training due to the inherent design of the reward formulation, without resorting to hand-tuned curricula or auxiliary stabilization tricks:

- **The "Too Hard" Regime (Low $\hat{p}(q)$):** If the model already struggles with the clean problem, a hinted failure provides weak evidence that the hint caused the disruption. The available positive credit is naturally throttled.
- **The "Too Easy" Regime (High $\hat{p}(q)$, High $\hat{p}(q,h_k)$):** If the Reasoner easily solves the clean problem and resists the hint, both rates are high, attenuating the update magnitude. This prevents a one-sided dominance collapse—a common self-play instability where the Adversary is over-penalized on questions that are too easy for the Reasoner.

Consequently, strong positive credit is exclusively concentrated on the intended sweet spot: questions that the Reasoner can solve cleanly but fails on when misled.

To optimize the Adversary, the number of hint candidates G_2 controls a trade-off between rollout cost and the resolution of within-question ranking. In cost-efficient settings—which we adopt as the default configuration for our main experiments—Seirênes applies a REINFORCE-style token update:

$$\mathcal{L}_{\text{Adv}} = -\frac{1}{G_2}\sum_{k=1}^{G_2} \text{sg}\Big[\hat{R}_{\text{Adv}}(q,h_k)\Big]\,\frac{1}{|h_k|}\sum_{t=1}^{|h_k|}\log \pi_\theta^{\text{Adv}}(h_{k,t} \mid q, h_{k,<t}), \tag{12}$$

where $h_{k,t}$ denotes the t-th token of hint h_k, and $\text{sg}[\cdot]$ denotes stop-gradient. While REINFORCE updates often suffer from high variance and instability, our formulation helps mitigate this issue. Rather than being a single-shot noisy reward, the advantage $\hat{R}_{\text{Adv}}(q,h_k)$ is derived from success rates averaged over G_1 and G_3 trajectories. This built-in variance reduction makes it feasible to treat the empirical degradation as a scalar advantage for updating the Adversary's generation probabilities.

Alternatively, when sampling a sufficiently large batch of hint candidates per question, the Adversary can use a grouped GRPO-style update to normalize within-question credit assignment. Here, we focus on the more accessible and effective default configuration.

Robustness branch. Finally, to reinforce correct reasoning under misleading context, this branch applies GRPO to the R3 trajectories. Let $\mathcal{H}_q \subseteq \{h_k\}_{k=1}^{G_2}$ denote the hint subset selected for the update, and let $\mathcal{Q}_{\text{rob}} = \{q : |\mathcal{H}_q| > 0\}$. The robustness objective is defined as:

$$\mathcal{L}_{\text{robust}} = \frac{1}{|\mathcal{Q}_{\text{rob}}|}\sum_{q\in\mathcal{Q}_{\text{rob}}}\frac{1}{|\mathcal{H}_q|}\sum_{h_k\in\mathcal{H}_q}\Big[-J_{\text{GRPO}}\Big(\theta; \{(q\oplus h_k, o_{k,j}^{(3)}, R_{k,j}^{(3)})\}_{j=1}^{G_3}\Big)\Big]. \tag{13}$$

The GRPO update is computed separately within each hint-conditioned group, while outer averages balance the signal across questions and their corresponding adversarial subsets.

Algorithm 1 Seirênes Training Orchestration

Require: Paired rollout bundle $\mathcal{D}$; pending queues $\{\mathcal{P}_s\}$; flush sizes $\{M_s\}$, with $s \in \{c, \mathrm{adv}, r\}$

Compute $\hat{p}(q)$, $\hat{p}(q, h_k)$, and $\hat{R}_{\mathrm{Adv}}(q, h_k)$ from $\mathcal{D}$.
Build candidate batches $\mathcal{C}_s$ and keep $\widetilde{\mathcal{C}}_s = \{g \in \mathcal{C}_s : \mathrm{A}_s(g)! = 0\}$ for $s \in \{c, \mathrm{adv}, r\}$.
for $s \in \{c, \mathrm{adv}, r\}$ **do**
  $\mathcal{P}_s \leftarrow \mathcal{P}_s \cup \widetilde{\mathcal{C}}_s$.
  if $|\mathcal{P}_s| \geq M_s$ **then**
    Update θ with $\mathcal{L}_s$ on $\mathcal{P}_s$; advance $\mathcal{P}_s$.
  end if
end for

4.3 Training orchestration

To preserve the distinct credit semantics of each role, Seirênes avoids naive batch pooling. Instead, it decouples the three branches into separate update streams, ensuring that gradients from clean reasoning do not interfere with those from adversarial generation or robustness recovery. Once a paired rollout bundle is evaluated, Seirênes constructs role-specific candidate batches. These candidates are then filtered by removing zero-advantage data ($\mathrm{A}_s(g)! = 0$) from each stream.

The valid data is accumulated in stream-specific pending queues until the corresponding flush size M_s is reached. As summarized in Algorithm 1, these queues act as cadence controllers: because clean, Adversary, and robustness groups survive filtering at different rates, the queues absorb these asymmetric arrival rates. An update to the shared policy is triggered only when a specific stream has gathered sufficient data to form a stable gradient estimate, after which the system advances the queue state. This orchestration ensures that each objective maintains its intended grouping while jointly optimizing the same underlying model. Long-horizon training dynamics and per-stream stability diagnostics are reported in Appendix G.

4.4 Efficient implementation

Three-round evaluation multiplies rollout costs. We therefore introduce practical strategies to maintain adversarial integrity while rendering overhead tractable.

Bounded FIFO buffers. To avoid the "reject-and-regenerate" waste common in online DAPO-style filtering (where valid data at batch boundaries is often discarded), the pending queues are implemented as role-specific Bounded FIFO (first in, first out) buffers. When a stream s reaches its flush size M_s, Seirênes consumes M_s groups for an update but crucially does not flush the entire buffer. Any remaining valid groups are carried over to subsequent steps; if a buffer reaches its capacity, Seirênes temporarily suspends sampling for that stream until the next update consumes the backlog. This is the physical realization of the advanced operation in Algorithm 1.

However, this persistence introduces data staleness as trajectories lag behind the latest policy. To mitigate off-policy drift, we discard stale rollout groups that lag significantly behind the current policy. This design differs from Selective Sample Replay [57] by prioritizing temporal recency for policy–data alignment rather than advantage magnitude for signal density.

Latency-aware rollout scheduling. Although our exposition introduces R_1, R_2, and R_3 as sequential stages for narrative clarity, only R_3 carries a true data dependency on the outputs of R_1 and R_2: both R_1 and R_2 condition solely on the original question q and impose no ordering constraint on each other. Seirênes exploits this slack by preconstructing the R_2 prompts and submitting them concurrently with the R_1 prompts to the rollout backend, collapsing the critical path to $T_{\mathrm{merged}} = T_{12} + T_{R_3}$. This merging is particularly effective because of the sequence length imbalance intrinsic to the RL loop: R_1 and R_3 are long reasoning trajectories that, as response lengths diverge within a batch, leave growing computational "bubbles" in the inference engine, whereas R_2 adversarial hints are markedly shorter. With continuous batching, the brief R_2 sequences are scheduled into these idle slots, so that the overhead of R_2 is effectively absorbed into the R_1 generation process and T_{12} tightly approximates T_{R_1}. The same schedule incidentally hides reward-computation latency: once R_2 hints are available, R_3 rollouts proceed immediately, and the R_1 reward verification is overlapped asynchronously with R_3 generation rather than sitting on the critical path.

Table 1: Results on mathematical reasoning benchmarks. Δ is the absolute improvement over the instruction-tuned baseline of the same model family. † Trained under the same budget as Seirênes.

Model	AIME24	AIME25	AIME26	IMO-Bench	MINERVA	Olympiad	HMMT26	AVG	ΔAVG
Qwen2.5-7B-Instruct	10.2	5.5	5.3	8.8	29.4	34.9	2.7	13.8	–
ExGRPO [73]	14.2	13.2	9.0	9.9	29.9	42.0	6.0	17.7	+3.9
LUFFY [65]	13.9	12.9	12.3	11.0	32.8	41.5	6.3	18.7	+4.9
SAGE [32]	15.8	13.7	9.3	11.2	31.6	42.2	6.3	18.6	+4.8
DAPO† [69]	14.1	12.0	9.3	9.5	30.5	42.1	4.3	17.4	+3.6
Seirênes (Ours)†	**22.8**	**15.4**	**17.8**	**13.3**	**34.7**	**48.0**	**8.5**	**22.9**	**+9.1**
Qwen3-4B-Instruct	61.5	48.8	56.8	34.1	35.4	66.3	31.9	47.8	–
Scaf-GRPO [77]	61.6	49.1	54.3	33.9	38.6	67.1	32.2	48.1	+0.3
SAGE [32]	60.1	49.1	57.8	32.5	41.1	67.2	29.9	48.2	+0.4
InT [68]	66.6	58.9	64.7	38.8	38.3	67.8	37.9	53.3	+5.5
Dr.GRPO† [38]	65.0	63.3	64.9	38.0	39.4	68.3	37.2	53.7	+5.9
DAPO† [69]	63.0	60.2	62.2	37.7	40.1	68.3	33.0	52.1	+4.3
Seirênes (Ours)†	**74.3**	**64.9**	**70.8**	**41.9**	**42.8**	**71.9**	**39.4**	**58.0**	**+10.2**
Qwen3-30B-A3B	75.0	60.6	75.1	40.5	35.0	70.0	41.0	56.7	–
DAPO† [69]	78.6	66.8	75.2	44.3	40.6	72.8	42.3	60.1	+3.4
Seirênes (Ours)†	**81.8**	**74.1**	**79.7**	**46.7**	**41.7**	**74.3**	**49.0**	**63.9**	**+7.2**

Mastery-aware sampling. As training progresses, certain problems become too easy for the Reasoner, solve cleanly even under adversarial hints. Continuing the full three-round loop on such instances yields diminishing learning signals while consuming a substantial rollout budget. We therefore maintain a mastered set $\mathcal{M}$ to deprioritize these questions in the active pool. While mastery-aware sampling is a general cost-control mechanism for RLVR, it is uniquely synergistic with our adversarial setting. By demanding robustness against targeted hints rather than mere clean-prompt success, it provides a more rigorous pruning signal, redirecting resources to instances that still expose model vulnerabilities. Formal operational criteria and the mastery indicator are detailed in Appendix D.

5 Experiments

5.1 Experimental setup

Benchmarks. We evaluate on seven mathematical reasoning benchmarks: AIME 2024–2026 [42, 43, 44], IMO-Bench [40], Minerva Math [29], OlympiadBench [15], and HMMT 2026 [4]. For AIME 2024–2026 and HMMT 2026 we report average accuracy across 32 independent samples per problem (avg@32); for MINERVA Math, OlympiadBench, and IMO-Bench we report avg@4.

Baselines. We compare against two main groups. *Standard RL baselines* establish the frontier without auxiliary context, including DAPO [69], ExGRPO [73] and Dr.GRPO [38]. *Hint-augmented baselines* inject auxiliary context into the RL loop with *cooperative* intent: SAGE [32] and Scaf-GRPO [77] use reference-guided scaffolds, LUFFY [65] relies on off-policy traces from stronger external models, and InT [68] relies on external references to localize reasoning errors. For compute-controlled comparisons, we train Seirênes and the DAPO/Dr.GRPO baselines under the same training budget; for other baselines, we use the released checkpoints when available.

Training and evaluation. We train Qwen2.5-7B-Instruct [67], Qwen3-4B-Instruct-2507 [66], and Qwen3-30B-A3B-Instruct-2507 [66] on 32 Ascend 910B NPU nodes. Training uses model-specific difficulty-balanced subsets from DAPO-Math-17K [69] and OpenR1-Math [22]. Unless otherwise stated, Seirênes uses $G_1{=}G_3{=}8$ Reasoner rollouts and $G_2{=}2$ adversarial hints per prompt; all evaluations use temperature 1.0 and top-p 1.0. See Appendix B for details.

5.2 Main results

Tab. 1 presents the clean, no-hint mathematical reasoning performance across three backbones. This evaluation tests whether the Seirênes framework—co-evolving a single policy as both Adversary and Reasoner—translates into stronger standalone reasoning ability in the base models. Across backbones, Seirênes improves upon the instruction-tuned baselines, showing positive gains across all evaluated datasets and raising the average accuracy. We highlight three patterns behind these results.

First, the improvements are not confined to weaker models. Qwen3-4B-Instruct starts from a higher baseline than Qwen2.5-7B-Instruct (47.8 vs. 13.8 AVG) yet achieves a higher gain (+10.2 vs. +9.1).

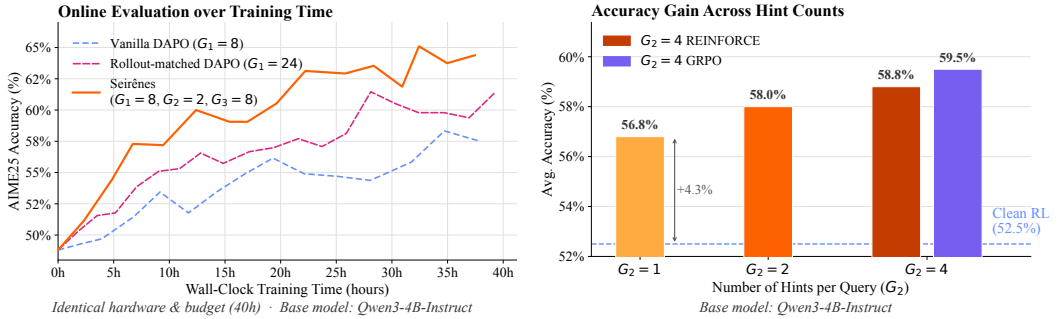

Figure 2: (a) AIME25 accuracy vs. wall-clock training time for three budget-matched conditions on identical hardware. (b) AVG accuracy vs. number of adversarial hints G_2 per step.

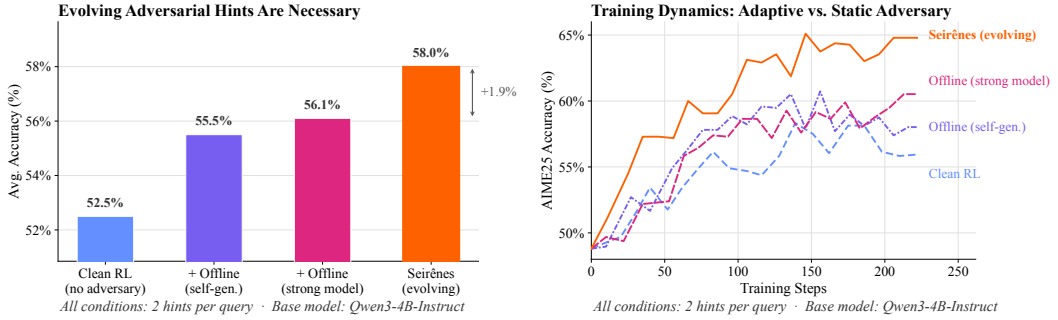

Figure 3: **Static vs. evolving adversarial hints.** (a) AVG accuracy across benchmarks for each condition. (b) AIME25 online accuracy over training steps.

This suggests that the adversarial loop remains useful beyond low-capacity initializations. The pattern also holds on a different architecture: on Qwen3-30B-A3B-Instruct, a sparse MoE backbone where RL can be less stable, Seirênes increases performance across the evaluation suite and outperforms the DAPO baseline by +3.8 AVG. Furthermore, the gains extend to benchmarks released after the base models' training cutoff (AIME 2026, HMMT 2026), where template memorization is less plausible.

Second, the gains persist across benchmarks that stress the models in different ways. Seirênes improves recent high-difficulty benchmarks such as AIME 2026 and HMMT 2026, while also improving broader mathematical reasoning evaluations such as IMO-Bench, MINERVA, and Olympiad-Bench. The effect is also visible on splits where the base model initially performs poorly: on Qwen2.5-7B-Instruct, HMMT 2026 rises from 2.7 to 8.5 and AIME 2026 from 5.3 to 17.8; on Qwen3-4B-Instruct, IMO-Bench moves from 34.1 to 41.9 and MINERVA from 35.4 to 42.8. Thus, the aggregate gains are not driven solely by benchmarks where the base model is already strong, but also appear on benchmarks where the base model has the most room to improve.

Third, cooperative-hint methods (LUFFY, SAGE, Scaf-GRPO, InT) show that auxiliary context is a productive ingredient for RLVR. Seirênes demonstrates competitive results using context in the opposite direction—adversarial rather than cooperative. This suggests that misleading context can serve as a training signal comparable to helpful scaffolding and that the adversarial loop itself is a viable axis for model improvement. Finally, the same-budget comparisons with DAPO and Dr.GRPO indicate that these gains are not simply recovered by allocating more compute to standard rollouts.

5.3 Analysis

Budget control. A natural concern is that Seirênes's gains might benefit from extra per-step compute introduced by R2 and R3 rollouts. We thus ablate three configurations on identical hardware under a fixed 40-hour wall-clock budget: vanilla DAPO with G_1=8 clean rollouts, rollout-matched DAPO with G_1=24 clean rollouts, and Seirênes with G_1=8, G_2=2, and G_3=8. The comparison isolates the source of the additional Reasoner rollouts: more clean rollouts versus G_2G_3 hint-conditioned rollouts driven by the Adversary. Fig. 2(a) plots AIME25 accuracy against wall-clock time.

Under this controlled comparison, Seirênes improves more rapidly throughout the 40-hour window. At matched wall-clock points, it reaches higher AIME25 accuracy earlier, indicating that the adver-

Table 2: Robustness to adversarial perturbations. Best per column in **bold**.

	Perturb Acc (↑)			Robustness Degradation (↓)			
Model	**Math-P-Hard**	**Math-P-Simple**	**Avg**	**GSMIR**	**MMLU-P**	**OBook-P**	**Avg**
Qwen3-4B-Instruct	73.9	93.9	83.9	**2.96**	9.39	19.03	10.46
Scaf-GRPO [77]	71.7	93.9	82.8	3.57	9.12	19.03	10.57
SAGE [32]	76.3	95.7	86.0	4.34	7.54	19.62	10.50
InT [68]	76.1	94.8	85.4	7.39	8.25	17.92	11.19
Seirênes (Ours)	**79.1**	**97.4**	**88.2**	3.05	**4.91**	**16.21**	**8.06**

sarially conditioned rollouts provide an effective training signal within the same compute envelope. Its endpoint is also the highest among the tested configurations, and the curve remains upward-moving at the end of the budget window. These trends support the view that Seirênes turns the additional rollout budget into faster and higher clean-reasoning improvement through adversarial co-evolution.

Effect of hint count. We vary the adversarial hint count ($G_2 \in \{1, 2, 4\}$, Fig. 2(b)) to test whether additional misleading contexts further improve training. Compared to a budget-matched baseline that omits the adversarial R_2 and R_3 rollouts, introducing even a single hint yields a substantial performance jump. Our default setting with G_2=2 further improves average accuracy to 58.0%. Scaling to G_2=4 achieves the highest absolute performance, indicating that the framework can continuously unlock further reasoning advantages when allocated a larger compute budget. While this trend suggests strong scaling potential, deploying $G_2 \geq 4$ until convergence significantly inflates the per-step inference cost and currently exceeds our computational resources. Fully exploring the upper bounds of this adversarial scaling remains a direction for future work.

Static vs. evolving adversarial hints. Beyond the mere volume of misleading contexts, we investigate whether static hints generated offline can reproduce the observed performance gains. To isolate this, we evaluate four configurations on the Qwen3-4B-Instruct (Fig. 3): standard clean RL, two static hint baselines, and Seirênes's evolving Adversary. Specifically, we evaluate *static self-generated hints* (produced offline by the Qwen3-4B-Instruct backbone itself using the identical adversarial prompt) and *static strong-model hints* (produced offline by Gemini 3 Flash [12]).

As illustrated in Fig. 3, while static hints provide a useful learning signal, they fail to match the efficacy of the adaptive loop. Notably, even the high-quality offline hints generated by Gemini 3 Flash plateau below Seirênes in overall accuracy. This contrast suggests that an adversarial signal is most effective when it dynamically tracks the Reasoner's evolving capabilities. Because static hints act as fixed snapshots of earlier distractors, they tend to become less effective at exposing the model's current failure modes once the Reasoner learns to bypass them. Seirênes, by refreshing the adversarial distribution online, keeps misleading contexts better aligned with the current decision boundary. Appendix E quantifies this effect with per-step attack-strength statistics and clean-validation traces. Empirically, dynamic co-evolution averts early saturation, yielding further performance gains.

Robustness to adversarial perturbations. We next investigate whether training against evolving adversarial context transfers beyond the training hint distribution by evaluating robustness under structural math perturbations and generated distractors across Math-Perturb [21], GSMIR [25], MMLU-Perturb [59], and OpenBookQA-Perturb [59]; benchmark details are deferred to Appendix C.

Tab. 2 demonstrates Seirênes's robustness across diverse perturbation families. On MATH-Perturb, Seirênes achieves the highest accuracy on both splits, improving the average perturbation accuracy from 83.9 for the instruction-tuned model to 88.2. The robustness-degradation columns further show that Seirênes is less easily distracted by generated irrelevant contexts, obtaining the lowest average degradation among all methods. Interestingly, the cooperative hint-based baselines all exhibit larger average degradation than the instruction-tuned model. While this does not establish a causal mechanism, it is consistent with the possibility that training with helpful auxiliary context can make the model more willing to trust externally provided cues, whereas Seirênes trains the model to scrutinize such cues before incorporating them into its reasoning. Together, these results suggest that adversarial-context training improves robustness in a way that is not limited to the specific negative-hint distribution used during training. Rather than merely reinforcing memorized templates or shallow pattern shortcuts, Seirênes appears to encourage reasoning behavior that better survives both structural reformulations and distracting contextual evidence.

Generalization to code. To test applicability beyond mathematics, we train Seirênes and clean GRPO from the same Qwen3-4B-Instruct-2507 initialization on Code-R1 LeetCode2k [36]. We use the original upstream split of 1,856 training and 200 held-out test problems, train both models for three epochs, and match the training questions, response budget, decoding configuration ($n = 32$), and SandboxFusion verifier. On the held-out test set, Seirênes improves pass@1 from 71.9% to 76.3% ($+4.4$ percentage points; 95% CI $[+2.1, +6.8]$) and pass@4 from 84.8% to 88.2% ($+3.4$ points; 95% CI $[+1.2, +5.7]$). These results provide initial evidence that Seirênes can extend to code reasoning in this setting. The complete results are reported in Appendix H.

Responsiveness to helpful context. We further examine whether Seirênes remains responsive to useful external information. On the pooled AIME 2024–2026 set (90 problems), we compare Seirênes and Qwen3-4B-Instruct using the same fixed hints generated by Claude Opus 4.8 from reference solutions. Helpful hints provide increasingly informative guidance: H1 suggests a relevant direction, H2 adds a key intermediate step, and H3 outlines a nearly complete solution without revealing the final answer. Both models are evaluated under identical inference settings using mean@32. As shown in Table 3, relative to its no-hint accuracy of 70.0%, Seirênes gains 1.1, 5.6, and 8.9 percentage points under H1, H2, and H3, respectively. Under misleading hints generated through the same protocol, its accuracy drops by 4.4 points, compared with 11.3 points for the base model. These results suggest that Seirênes retains the ability to benefit from helpful context while exhibiting reduced susceptibility to misleading information.

Table 3: Accuracy with misleading and helpful hints on pooled AIME 2024–2026 ($n = 90$, mean@32). Parentheses: changes from no-hint accuracy in percentage points.

Dataset	Model	No hint	Adversarial	Helpful H1	Helpful H2	Helpful H3
AIME24–26 pooled	Qwen3-4B-Instruct	55.7	44.4 (-11.3)	56.7 ($+1.0$)	60.0 ($+4.3$)	66.7 ($+11.0$)
AIME24–26 pooled	Seirênes	70.0	65.6 (-4.4)	71.1 ($+1.1$)	75.6 ($+5.6$)	78.9 ($+8.9$)

What does the Adversary learn to say? Representative examples and diagnostics in Appendix F show that the learned hints typically take the form of *half-true* mathematical continuations, exploiting plausible reasoning shortcuts rather than injecting random noise.

To evaluate these hints on other reasoners, we use the Seirênes Adversary trained on top of Qwen3-4B-Instruct to offline-generate one hint per problem. We then append these hints to the original problem descriptions and prompt GPT-5.1 and Gemini 3 Flash to solve the resulting hinted problems. We measure the resulting performance degradation using avg@3 accuracy across seven benchmarks.

Tab. 4 provides diagnostic evidence of cross-model attack effect. Exposure to the 4B-generated hints reduces the accuracy of both proprietary targets across all evaluated domains. The degradation is especially pronounced on competition-level benchmarks: Gemini 3 Flash drops by 10.0 absolute points on AIME26 and 12.9 points on IMO-Bench, despite having much stronger plain accuracy than GPT-5.1. This suggests that the learned hints are not merely overfitted to the training Reasoner; they encode misleading reasoning patterns that also work on stronger proprietary reasoners.

Table 4: Attack effectiveness of the Seirênes Adversary (4B) on external proprietary models. $\mathbf{\Delta}$: hinted accuracy minus plain accuracy (negative = degraded). Avg@3 reported for all benchmarks.

Model	AIME24	AIME25	AIME26	IMO-Bench	Math-P-Hard	MINERVA	Olympiad	Overall
GPT-5.1 (Clean)	43.3	36.7	43.3	15.3	65.2	21.3	52.2	39.6
GPT-5.1 (Hinted)	36.7	30.0	40.0	13.3	60.6	19.9	49.2	35.7
$\mathbf{\Delta}$	-6.6	-6.7	-3.3	-2.0	-4.6	-1.4	-3.0	-3.9
Gemini 3 Flash (Clean)	90.0	90.0	93.3	67.2	82.6	43.8	81.8	78.4
Gemini 3 Flash (Hinted)	84.4	86.7	83.3	54.3	80.0	42.2	81.2	73.2
$\mathbf{\Delta}$	-5.6	-3.3	-10.0	-12.9	-2.6	-1.6	-0.6	-5.2

6 Conclusion

We introduce Seirênes, a shared-parameter self-play RL framework that turns contextual interference into an internal training signal for mathematical reasoning. A single policy co-evolves as both an Adversary that generates plausible but distracting contexts and a Reasoner that recovers the core problem logic under such interference. Across model scales and seven mathematical benchmarks, Seirênes yields consistent gains for standalone reasoning ability.

Acknowledgments and Disclosure of Funding

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0123602).

The numerical calculations in this paper have been done in part on the supercomputing system in the Supercomputing Center of Wuhan University.

The authors declare no additional competing interests.

References

[1] Chen Amiraz, Florin Cuconasu, Simone Filice, and Zohar Karnin. The distracting effect: Understanding irrelevant passages in rag. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18228–18258, 2025.

[2] Lei Bai, Zhongrui Cai, Yuhang Cao, Maosong Cao, Weihan Cao, Chiyu Chen, Haojiong Chen, Kai Chen, Pengcheng Chen, Ying Chen, et al. Intern-s1: A scientific multimodal foundation model. *arXiv preprint arXiv:2508.15763*, 2025.

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[4] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025.

[5] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 23–42. IEEE, 2025.

[6] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *AAAI Conference on Artificial Intelligence*, 2024.

[7] Kun Chen, Peng Shi, Fanfan Liu, Haibo Qiu, Zhixiong Zeng, Siqi Yang, and Wenji Mao. Flexible entropy control in RLVR with a gradient-preserving perspective. *arXiv preprint arXiv:2602.09782*, 2026.

[8] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[10] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

[11] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

[12] Gemini Team. Gemini 3 flash: frontier intelligence built for speed, 2025.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, et al. Justrl: Scaling a 1.5 b llm with a simple rl recipe. *arXiv preprint arXiv:2512.16649*, 2025.

[15] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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 (Volume 1: Long Papers)*, pages 3828–3850, 2024.

[16] Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. Visplay: Self-evolving vision-language models from images. *arXiv preprint arXiv:2511.15661*, 2025.

[17] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[18] Homer. *The Odyssey*. Oxford World's Classics. Oxford University Press, Oxford, 2008. Translated by Walter Shewring; introduction by G. S. Kirk.

[19] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.

[20] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

[21] 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.

[22] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.

[23] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[24] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 2021–2031, 2017.

[25] Ming Jiang, Tingting Huang, Biao Guo, Yao Lu, and Feng Zhang. Enhancing robustness in large language models: Prompting for mitigating the impact of irrelevant information. In *International Conference on Neural Information Processing*, pages 207–222. Springer, 2024.

[26] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[27] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[28] Seongyun Lee, Yongrae Jo, Minju Seo, Moontae Lee, and Minjoon Seo. Lost in the noise: How reasoning models fail with contextual distractors. *arXiv preprint arXiv:2601.07226*, 2026.

[29] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

[30] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

[31] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.

[32] Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026.

[33] Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

[34] Fanfan Liu, Youyang Yin, Peng Shi, Siqi Yang, Zhixiong Zeng, and Haibo Qiu. Length-unbiased sequence policy optimization: Revealing and controlling response length variation in RLVR. *arXiv preprint arXiv:2602.05261*, 2026.

[35] Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning. *arXiv preprint arXiv:2511.19900*, 2025.

[36] Jiawei Liu and Lingming Zhang. Code-R1: Reproducing R1 for code with reliable rewards. `https://github.com/ganler/code-r1`, 2025.

[37] Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025.

[38] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[39] Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, et al. Deepcoder: A fully open-source 14b coder at o3-mini level. *Notion Blog*, 1, 2025.

[40] Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35406–35430, 2025.

[41] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[42] Mathematical Association of America. American invitational mathematics examination (aime) 2024. `https://maa.org`, February 2024. Competition problems and official solutions.

[43] Mathematical Association of America. American invitational mathematics examination (aime) 2025. `https://maa.org`, February 2025. Competition problems and official solutions.

[44] Mathematical Association of America. American invitational mathematics examination (aime) 2026. `https://maa.org`, February 2026. Competition problems and official solutions.

[45] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2381–2391, 2018.

[46] Seyed Mahed Mousavi, Edoardo Cecchinato, Lucia Hornikova, and Giuseppe Riccardi. Garbage in, reasoning out? why benchmark scores are unreliable and what to do about it. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 1747–1759, 2026.

[47] OpenAI Team. Gpt-5.1: A smarter, more conversational chatgpt, 2025.

[48] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, 2022.

[49] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning*, pages 2817–2826. PMLR, 2017.

[50] Haibo Qiu, Xiaohan Lan, Fanfan Liu, Xiaohu Sun, Delian Ruan, Peng Shi, and Lin Ma. Metis-RISE: RL incentivizes and SFT enhances multimodal reasoning model learning. *arXiv preprint arXiv:2506.13056*, 2025.

[51] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[52] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[53] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[54] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[55] Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

[56] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 2153–2162, 2019.

[57] Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *arXiv preprint arXiv:2504.08837*, 2025.

[58] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[59] Yanbo Wang, Zixiang Xu, Yue Huang, Chujie Gao, Siyuan Wu, Jiayi Ye, Pin-Yu Chen, Xiuying Chen, and Xiangliang Zhang. Adaptive distraction: Probing llm contextual robustness with automated tree search. *arXiv preprint arXiv:2502.01609*, 2025.

[60] Yuqing Wang and Yun Zhao. Rupbench: Benchmarking reasoning under perturbations for robustness evaluation in large language models. *arXiv preprint arXiv:2406.11020*, 2024.

[61] Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. How easily do irrelevant inputs skew the responses of large language models? *arXiv preprint arXiv:2404.03302*, 2024.

[62] Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

[63] Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026.

[64] Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 2742–2785, 2025.

[65] Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

[66] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[67] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[68] Matthew YR Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. Int: Self-proposed interventions enable credit assignment in llm reasoning. *arXiv preprint arXiv:2601.14209*, 2026.

[69] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[70] Tong Yu, Yongcheng Jing, Xikun Zhang, Wentao Jiang, Wenjie Wu, Yingjie Wang, Wenbin Hu, Bo Du, and Dacheng Tao. Benchmarking reasoning robustness in large language models. *arXiv preprint arXiv:2503.04550*, 2025.

[71] Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. Guided self-evolving llms with minimal human supervision. *arXiv preprint arXiv:2512.02472*, 2025.

[72] Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

[73] Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*, 2025.

[74] Chi Zhang, Haibo Qiu, Qiming Zhang, Yufei Xu, Zhixiong Zeng, Siqi Yang, Peng Shi, Lin Ma, and Jing Zhang. Perceptual-evidence anchored reinforced learning for multimodal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 41111–41120, 2026.

[75] Chi Zhang, Haibo Qiu, Qiming Zhang, Zhixiong Zeng, Lin Ma, and Jing Zhang. DeepSketcher: Internalizing visual manipulation for multimodal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pages 9389–9400, 2026.

[76] Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. Stephint: Multi-level stepwise hints enhance reinforcement learning to reason. *arXiv preprint arXiv:2507.02841*, 2025.

[77] Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. *arXiv preprint arXiv:2510.19807*, 2025.

[78] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

[79] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[80] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A Extended related work

Adversarial examples for language understanding. Text-level adversarial inputs are a long-standing tool for exposing brittleness in neural NLP models. Jia and Liang [24] show that appending semantically related but task-irrelevant sentences to reading-comprehension passages causes sharp accuracy drops, a phenomenon directly mirrored at the LLM scale by GSM-IC [53], which demonstrates that grade-school math problems can be derailed simply by inserting irrelevant quantitative sentences. Wallace et al. [56] further show that short, input-agnostic trigger strings transfer across both inputs and target models, foreshadowing the cross-model attack pattern we observe in Section 5.3, where a 4B Seirênes Adversary degrades GPT-5.1 [47] and Gemini-3-Flash [12]. These works share Seirênes's core threat model of using plausible text to derail reasoning, yet they relegate adversarial context strictly to evaluation, testing static reasoners against fixed attacks. Seirênes fundamentally inverts this approach. Instead of treating adversarial vulnerability as a static metric, it integrates the attacker into the RL loop, co-evolving the Reasoner against a dynamic Adversary to turn a traditional vulnerability into a constructive training signal.

Adversarial training and co-evolving RL adversaries. The constructive use of adversaries during training finds its canonical form in Madry et al. [41], which formalizes adversarial training as a saddle-point minimax optimization with an inner PGD attacker producing norm-bounded input perturbations. Pinto et al. [49] (RARL) lift this concept to reinforcement learning by training a protagonist policy alongside a disturbance adversary that perturbs environment dynamics. PAIRED [10] pushes this further by having an adversary generate regret-maximizing environments, creating an autocurriculum that continuously challenges the agent at its capability frontier. Seirênes inherits this minimax lineage but introduces three substantive shifts. *First*, the perturbation takes the form of a coherent natural-language hint rather than a norm-bounded token manipulation or a scalar control disturbance. This shifts the attack surface from gradient-based token optimization to the semantic reasoning level. *Second*, the core components of the environment, including the question, the verifier, and the action space, remain strictly unchanged. Only the input *context* is perturbed, ensuring the task remains well-specified while still posing a rigorous challenge. *Third*, the Adversary and the Reasoner share parameters and are differentiated solely by role conditioning. This design collapses the separate networks typically required in RARL or PAIRED into a single update flow, avoiding a twofold increase in parameter count.

LLM red-teaming and shared-parameter self-play. A parallel line of work uses adversarial probing to characterize and improve safety-aligned LLMs. Perez et al. [48] and Ganguli et al. [11] prompt or train language models to attack other language models, surfacing harmful behaviors at scale; GCG [80] and PAIR [5] construct adversarial prompts via gradient search and iterative black-box querying, respectively, to bypass safety alignment. Constitutional AI [3] and SPIN [8] further illustrate that a single language model can play two roles via shared parameters—critic versus actor in the former, and weak-data-generator versus stronger-target in the latter. Seirênes adopts the shared-parameter trick from this lineage but differs in two essential respects. The optimization target is *reasoning capability* on a verifiable task rather than safety alignment or weak-to-strong distillation, so the adversarial signal must take the form of mathematically plausible derivations rather than alignment-bypassing prompts. And the use of attacks is fully in-loop: adversarial hints are produced by the same policy being optimized and consumed within the same RL update, rather than serving as one-shot evaluation probes [48, 80, 5], a fixed dataset for downstream supervised tuning [11], or a cooperative critic-actor arrangement aimed at distributional alignment [3, 8]. To our knowledge, Seirênes is the first framework to combine all three properties—a learned natural-language attacker, a shared-parameter self-play loop, and a verifiable reasoning objective—under a single online RL update.

Reasoning Robustness under Contextual Perturbations. Despite the striking capability gains driven by RLVR and self-evolving training, reasoning LLMs often exhibit brittleness when operating outside the pristine conditions of standard benchmarks. Recent evaluations including MATH-Perturb, RUPBench, and Math-RoB report severe degradation under structural reformulations, semantic perturbations, and incomplete information [21, 60, 70]. Concurrently, complementary analyses highlight memorization, format sensitivity, and a reliance on superficial cues as persistent confounders in assessing true reasoning capabilities [64, 46]. Beyond structural shifts, contextual distraction poses another significant challenge: biasing features can misdirect chain-of-thought rationale [55], while task-irrelevant but semantically related contexts severely impair reasoning perfor-

mance [59, 61]. These vulnerabilities frequently surface in real-world deployment scenarios plagued by noisy retrieval documents, hard distractors, and erroneous tool outputs [6, 1, 28].

B Experimental reproducibility details

Training backbones and infrastructure. We train on three backbones of varying scale: Qwen2.5-7B-Instruct [67], Qwen3-4B-Instruct-2507 [66], and Qwen3-30B-A3B-Instruct-2507 [66]. All models are trained using verl [52] with vLLM [26] for rollout generation. The Adversary and Reasoner share a single set of parameters and are separated only through role-conditioned prompting. The prompt template is shown in Appendix K.

Training data construction. For training data, we pool problems from DAPO-Math-17K [69] and OpenR1-Math [22]. To ensure that the RL process is supported by a well-distributed dataset optimally aligned with each model's individual capacity, we construct a tailored subset for every backbone. Concretely, for each backbone, we run an offline pass with 8 rollouts per problem at temperature 1.0, use the empirical pass rate to estimate difficulty, and then perform balanced sampling across difficulty buckets.

Optimization hyperparameters. We use AdamW with a constant learning rate of 1×10^{-6}, no warm-up, KL penalty disabled ($\beta=0$), and asymmetric clipping with $\varepsilon_{\text{low}}=0.2$ and $\varepsilon_{\text{high}}=0.28$ [69]. Each rollout batch contains 256 prompts with $G_1=G_3=8$ responses per prompt and a maximum response length of $16{,}384$ tokens. By default, the Adversary generates $G_2=2$ misleading hints per prompt; we ablate this choice in Section 5.3. All evaluation inference uses temperature 1.0 with top-p 1.0.

Stream scheduling and auxiliary losses. For all main runs, we set the stream flush sizes to $M_c=128$ clean prompt groups ($128 \times 8=1024$ trajectories), $M_{\text{adv}}=256$ R2 adversarial hint trajectories, and $M_r=128$ hint-conditioned Reasoner groups (1024 trajectories). Stale buffered entries are evicted after at most 3 optimizer-step lag for the clean stream, 3 optimizer-step lag for the Adversary stream, and 3 optimizer-step lag for the robustness stream. The R2 adversarial hint generation is capped at 2048 tokens.

Training length. For the main results in Tab. 1, all three backbones use the same nominal training schedule of 200 rollout-collection steps. Because Seirênes uses stream-specific FIFO buffers and branch-specific flush cadences, the actual number of steps can fluctuate slightly around this nominal step count across backbones and streams.

C Robustness evaluation details

Benchmark families. We evaluate robustness beyond the training hint distribution using two complementary perturbation families. The first family tests robustness to structural reformulations of math problems. MATH-Perturb-Simple and MATH-Perturb-Hard modify competition-level math problems through structural and numerical perturbations [21], testing whether the model can preserve mathematical reasoning when the problem formulation changes. Because these splits remain math-domain evaluations, we report absolute perturbed accuracy.

The second family tests robustness to generated distractors. GSMIR [25] augments GSM8K-style [9] elementary math word problems with semantically related but irrelevant sentences. MMLU-Perturb [59] and OpenbookQA-Perturb [59] provide out-of-domain evaluations by injecting answer-preserving distractors into multiple-choice problems. MMLU [17] spans broad academic subjects across STEM, humanities, and social sciences, while OpenbookQA [45] requires applying elementary-science facts together with commonsense reasoning.

Perturbation generation and metrics. For all benchmarks involving generated distractors, we use Gemini 3 Flash to produce the perturbing contexts. The prompts used for test-time perturbation generation are independent of the prompts used during Seirênes training. For GSMIR, MMLU-Perturb, and OpenbookQA-Perturb, we report robustness degradation rather than absolute perturbed accuracy, where lower degradation indicates that the model is less affected by the added irrelevant context.

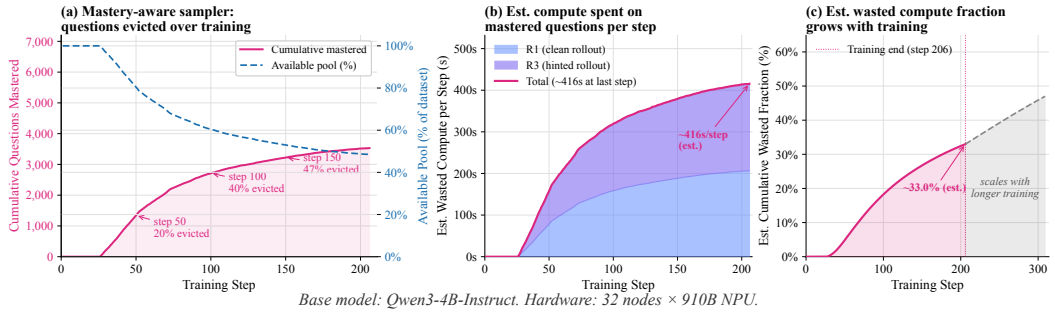

Figure 4: **Mastery-aware sampler: eviction dynamics and estimated compute savings** (Qwen3-4B-Instruct, $|\mathcal{D}|$=6,853). **(a)** Cumulative mastered set (left axis) and live adversarial pool fraction (right axis) over training. **(b)** Per-step compute estimated to be spent on already-mastered questions, broken down by R1 (clean) and R3 (hinted) rollouts. **(c)** Cumulative wasted fraction over training, with a logistic extrapolation (dashed; fit on observed steps 1–206) illustrating how the fraction grows with horizon. All estimates use a linear upper-bound model; actual savings are strictly smaller because rollout batch time is dominated by the longest trajectory.

D Mastery-aware sampling

Motivation and mechanism. As training progresses, some questions become too easy for the current Reasoner: it solves them cleanly, and the current Adversary's evaluated hints no longer induce failures. Continuing to run the full R1–R3 loop on such questions consumes rollout budget while producing little useful learning signal. The mastery-aware sampler therefore maintains an operational mastered set $\mathcal{M}$ and deprioritizes these questions in the live adversarial pool.

At evaluation step τ, define

$$m_\tau(q) = \mathbf{1}\left[\hat{p}_\tau(q) = 1 \ \wedge \ \forall h_k \in \mathcal{H}_q^{(\tau)},\ \hat{p}_\tau(q, h_k) = 1\right], \tag{14}$$

where $\hat{p}_\tau(q)$ and $\hat{p}_\tau(q, h_k)$ denote the estimators in Eqs. 5 and 8 computed from the rollouts collected at step τ, $\mathcal{H}_q^{(\tau)}$ is the set of hinted groups for question q that survive the zero-signal filter (Section 4.3) at step τ, and we evaluate $m_\tau(q)$ at every training step using the rollout statistics produced at that step. This is an operational criterion, not a claim of permanent mastery: q enters $\mathcal{M}$ only once $m_\tau(q) = 1$ holds for K_m consecutive evaluation steps (where $K_m = 1$ in our experiment). The aggressive choice $K_m = 1$ is empirically reliable because the joint all-correct condition over G_1 clean and $\sum_k G_3$ hinted rollouts already provides strong implicit averaging at a single step; the post-hoc audit in Tab. 5 confirms that the resulting retired set has 98.9% mean@8 with no question below $4/8$. While mastery-aware sampling can be viewed as a general cost-control mechanism for RLVR training, it is especially aligned with our adversarial setting because retirement requires robustness to targeted hints rather than clean-prompt success alone.

What does mastery-aware sampling actually buy us? The practical value of mastery-aware sampling stems from two coupled effects: computational efficiency and eviction reliability. First, because Seirênes's three-round paired rollout incurs a per-question overhead, retiring robust questions directly reduces wasted compute. Second, because eviction relies on success under adversarial interference, the sampler should ideally retire only genuinely mastered questions. We evaluate the sampler along these two axes.

To quantify compute savings, we track the cumulative mastered set on the 6,853-question Qwen3-4B-Instruct pool. We estimate the theoretical saved rollout time via a linear projection, assuming rollout work scales proportionally with the active pool size:

$$\widehat{T}_{\text{saved}}(s) \ = \ \tfrac{|\mathcal{M}_s|}{|\mathcal{D}|} \cdot \left(\bar{T}_{R1} + G_2 \bar{T}_{R3}\right),$$

where $\bar{T}_{R1}{\approx}401\,\text{s}$ and $\bar{T}_{R3}{\approx}406\,\text{s}$ are the average rollout times measured in a $G_2{=}1$ probe run on the same backbone and infrastructure. The reported percentage saving is invariant to G_2 because both the retired and active per-step compute scale by the same factor $\bar{T}_{R1} + G_2\,\bar{T}_{R3}$; we therefore use these $G_2{=}1$ timings as a representative scale.

To assess eviction reliability, we conduct a post-hoc audit by re-evaluating each retired question with $n = 8$ clean rollouts at temperature 1.0. We compare this accuracy distribution against a matched DAPO sampler keyed on clean-success retirement. Both samplers share identical K_m requirements, evaluation cadence, and step horizon. The only difference is that its mastery indicator weakens our criterion $\hat{p}_\tau(q)=1 \wedge \forall h_k \in \mathcal{H}_q^{(\tau)}, \hat{p}_\tau(q,h_k)=1$ to clean-success alone, $\hat{p}_\tau(q)=1$.

Results. The cumulative mastered set grows rapidly: 1,339 questions (20%) by step 50, 2,713 questions (40%) by step 100, and 3,531 questions (52%) by step 206 (Fig. 4). Under the linear projection, the sampler reclaims approximately 33.9% of the total training compute, with the per-step reduction reaching 52% by the end of training.

Post-hoc verification confirms that our hint-conditioned threshold identifies a higher-confidence mastered set than clean-success retirement alone. As shown in Tab. 5, Seirênes's retired questions achieve 98.9% mean@8, compared to 95.5% for the matched DAPO sampler. They also obtain a higher all-correct rate (93.0% vs. 89.4%) and contain no questions below $4/8$ correct, compared to 1.2% for DAPO. Ultimately, these findings suggest that our approach achieves a robust reduction in computational cost without prematurely discarding valuable training signals.

Table 5: Post-hoc verification of retired questions (n=8 clean rollouts, temperature 1.0).

Model	Audit rollouts	mean@8	pass@8	8/8	7/8	<4/8
DAPO [69]	29,928	95.5%	100.0%	89.4%	3.4%	1.2%
Seirênes (Ours)	28,592	98.9%	100.0%	93.0%	5.8%	0.0%

E Data-level evidence for hint staleness

The static-hint ablation in Fig. 3 shows that offline adversarial contexts improve learning but saturate earlier than Seirênes. Here we provide a data-level explanation grounded in the per-step attack strength logged on the Qwen3-4B-Instruct backbone. Building on the per-question success rates $\hat{p}(q)$ and $\hat{p}(q,h)$ from Eqs. 5 and 8, let $\bar{p}_1(s) = \mathbb{E}_q[\hat{p}_s(q)]$ and $\bar{p}_3(s) = \mathbb{E}_{q,h}[\hat{p}_s(q,h)]$ denote their batch-mean values at training step s on the clean prompt and on the same batch under hint conditioning, respectively. We use

$$\Delta_{\text{attack}}(s) \;=\; \bar{p}_1(s) - \bar{p}_3(s)$$

as an operational proxy for the realized attack pressure exerted by the current hint distribution on the current Reasoner. Because $\bar{p}_1, \bar{p}_3 \in [0,1]$ are pass rates, Δ_{attack} is a difference in success rate that we report in percentage points. All statistics below are computed on the common training range (steps 1–206) shared by Seirênes and the two static baselines. We define strong-attack steps by $\Delta_{\text{attack}}(s) > 5\%$. The 21-step smoothing window used below corresponds to approximately 10% of this common training horizon, providing a fixed-scale trend estimate rather than a tuned window.

Table 6: Per-step attack-strength statistics on Qwen3-4B-Instruct over the common training range (steps 1–206). Tail frequency reports $\Pr(\Delta_{\text{attack}} > \theta)$ at different thresholds. Saturation splits strong-attack steps, defined by $\Delta_{\text{attack}} > 5\%$, between the early and late halves of training; slope is computed from the 21-step moving average of this strong-attack indicator.

	Tail frequency $\Pr(\Delta_{\text{attack}} > \theta)$ (%)			**Saturation** $\Pr(\Delta_{\text{attack}} > 5\%)$ (%)		**Slope**	**Sustained pressure**	
Run	θ=3%	θ=4%	θ=5%	Early (1–103)	Late (104–206)	(%/step)	Longest streak	Total strong steps
Static self	36.4	18.0	6.3	8.7	3.9	−0.012	1	13
Static strong (Gemini 3 Flash)	30.1	11.2	2.4	3.9	1.0	−0.007	1	5
Seirênes (Ours)	**55.8**	**34.5**	**19.4**	**18.4**	**20.4**	**+0.057**	**11**	**40**

Tail amplification. The gap between Seirênes and the static baselines *grows* as the threshold θ tightens. At θ=3% Seirênes (55.8%) is 1.5–1.9× the static methods (36.4% / 30.1%); at θ=5% the ratio widens to 3.1–8.1× (19.4% vs. 6.3% / 2.4%). The conclusion is robust to threshold choice, and the relative gap grows with stricter thresholds, indicating that the gap lives in the strong-attack tail rather than the bulk.

Saturation reversal. Static-self halves its strong-attack ratio between the early and late halves of training (8.7% → 3.9%) and static-strong nearly vanishes (3.9% → 1.0%); Seirênes *increases*

($18.4\% \rightarrow 20.4\%$). A linear regression of the 21-step rolling strong-attack ratio against training step gives slopes -0.012%/step for static-self, -0.007%/step for static-strong, and $+0.057\%$/step for Seirênes — the static methods' adversarial pressure decays over training while Seirênes's *strengthens*.

Sustained pressure. Seirênes produces a 40-step strong-attack budget across the run with a longest run-length of *11 consecutive steps*. The two static baselines deliver only 5–13 strong-attack steps in total, and *never* sustain pressure beyond a single step (longest streak $= 1$). Under this criterion, strong static attacks therefore appear only as isolated events, whereas strong Seirênes attacks come in sustained bursts that exert prolonged learning pressure on the Reasoner.

These three properties — broader tail with a relative gap that grows under stricter thresholds, opposite-sign saturation trajectory, and an order-of-magnitude longer sustained-attack streak — supply a coherent data-level mechanism for the clean-validation gap reported in Fig. 3. Both static baselines use a fixed offline hint distribution generated before training and replayed throughout training; as the Reasoner learns to bypass this non-adaptive distribution of distractors, the same hints become increasingly off-frontier and the strong-attack tail thins out. Seirênes refreshes the adversarial distribution online, keeps the hint generator on the Reasoner's current decision boundary, and preserves a substantially thicker strong-attack tail throughout training.

F Hint effectiveness

The cross-model evaluation is reported in Section 5.3; here we provide the detailed qualitative analysis and references to representative cases.

The anatomy of the generated traps. To understand the source of this transfer effect, we conduct a qualitative inspection of hints sampled from the diagnostic experiment in Section 5.3. Representative examples of the generated hints and the resulting model trajectories are provided in Section J. The analysis confirms that the Adversary does not rely on random noise. Instead, it converges on a "Trojan horse" strategy: synthesizing *half-true*, mathematically plausible derivations. These hints typically begin with a valid or familiar first step, establishing traction with a reasoning model's priors, and then quietly insert a fatal logical fallacy or unjustified leap.

We observe distinct manifestations of this deception across mathematical domains. In geometry problems, the hint frequently turns valid local equal-area or equal-angle facts into false global symmetry or closure assumptions. In algebraic problems, it often disguises invalid variable cancellations or unjustified equality bounds within otherwise standard algebraic simplifications. In number-theoretic and combinatorial problems, it subtly replaces exact residue calculations or rigorous case analysis with false uniformity assumptions or invalid lifting shortcuts. These locally correct initial moves successfully provide camouflage for the later invalid leaps, explaining why the learned hints remain highly effective against capable proprietary reasoners on multi-step benchmarks such as AIME and IMO-Bench. Rather than failing at superficial prompt attacks, the target models are baited into extending the Adversary's flawed premises into incorrect final answers, demonstrating that the Adversary has genuinely learned to construct structured mathematical traps.



Figure 5: Extended training dynamics on Qwen2.5-7B-Instruct. Validation accuracy improves over the prolonged trace, while the hinted-clean gap remains negative.

G Extended training dynamics and stability diagnostics

Self-improvement methods trained on model-generated signals can become unstable over long horizons: the training signal may vanish, the Adversary may collapse, or repeated policy updates may

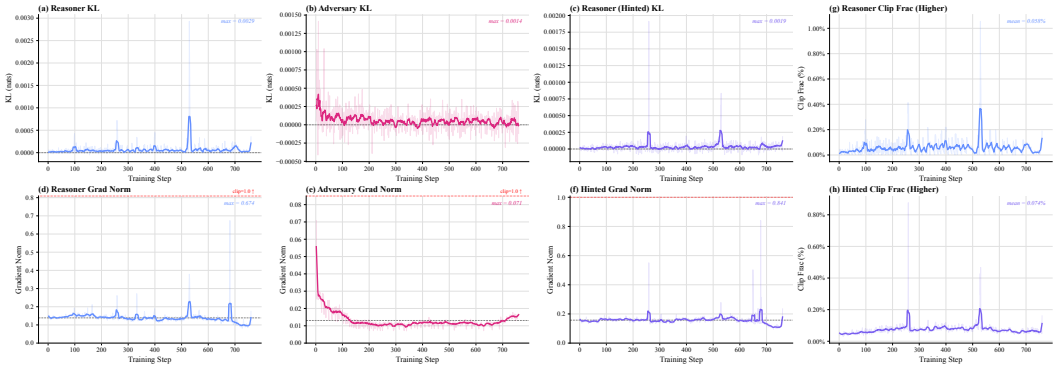

Figure 6: Stability diagnostics for the same extended Qwen2.5-7B-Instruct run. PPO KL estimates and gradient norms remain bounded across branches, with near-zero clip fractions for the Reasoner updates.

produce large KL or gradient spikes. We therefore include an extended diagnostic run on Qwen2.5-7B-Instruct, which provides a roughly $3\times$ longer training horizon than the main reported training traces. The x-axis counts logged scheduler iterations; due to branch-specific buffering and flushing, this trace corresponds to 699 main updates and roughly 2.1K actor update calls across the R1, R2, and R3 branches.

Fig. 5 shows that Seirênes continues to make progress over the extended trace. Both AIME25 and AIME26 online validation accuracy improve despite the run being substantially longer than the primary training traces. At the same time, the clean Reasoner reward remains above the hint-conditioned Reasoner reward, and the hinted-clean gap stays negative. This suggests that the Adversary continues to find misleading contexts that stress the current Reasoner, rather than exhausting its useful attack signal early in training. The entropy curves provide a complementary view of the same co-evolutionary process. We observe that the Adversary branch operates at higher entropy than the two Reasoner branches, an ordering that follows naturally from the asymmetry between solving and adversarial hinting. The Reasoner branches instantiate a behavior that is already familiar to an instruction-tuned model: solving a math problem under the given context. Their successful trajectories are also tightly constrained by the verifier and the unique correct answer. In contrast, the Adversary searches over plausible but misleading mathematical contexts. This adversarial role is less directly aligned with the standard instruction-following prior and has a broader effective semantic search space, since there are many more ways to be plausibly wrong than to follow a correct solution path. Thus, the higher Adversary entropy reflects continued exploration over diverse misleading strategies. The persistent separation between clean and hint-conditioned rewards further shows that these strategies remain strong enough to depress the Reasoner's performance under adversarial context throughout the extended run.

Fig. 6 further reports optimization stability diagnostics for the three update streams. The PPO KL estimates remain small: the maximum observed values are 2.9×10^{-3} for R1, 1.4×10^{-3} for R2, and 1.9×10^{-3} for R3, with 95th percentiles below 2.7×10^{-4}. Gradient norms also stay below 0.85 across all three branches, and the reported clip fractions for the two Reasoner branches average only 0.058% and 0.074% for R1 and R3, respectively.

Together, these results support the interpretation that Seirênes maintains a persistent adversarial training signal over a prolonged horizon while keeping the three update streams numerically stable. We use these curves as diagnostics of long-horizon training behavior, rather than as a separate backbone comparison.

H Code-domain generalization

We train Seirênes and clean GRPO from the same Qwen3-4B-Instruct-2507 initialization on Code-R1 LeetCode2k [36], using the original upstream split of 1,856 training and 200 held-out test problems. Both methods are trained for three epochs. We match the training questions, response budget, decoding configuration ($n = 32$), and SandboxFusion verifier. Table 7 reports the held-out pass@1 and pass@4 results and the 95% confidence intervals for the differences between Seirênes and GRPO.

Table 7: Held-out code-generation performance on Code-R1 LeetCode2k. Differences and their 95% confidence intervals are in percentage points.

Metric	GRPO 3 ep	Seirênes 3 ep	Δ vs. GRPO 3 ep	95% CI
pass@1	71.9	76.3	+4.4	[+2.1, +6.8]
pass@4	84.8	88.2	+3.4	[+1.2, +5.7]

I Data contamination

Our evaluation suite is deliberately weighted toward recent benchmarks, including several contests released after the public release of the relevant instruction-tuned backbones, providing a structural safeguard against data contamination. Qwen2.5-7B-Instruct was publicly released in September 2024, whereas Qwen3-4B-Instruct-2507 and Qwen3-30B-A3B-Instruct-2507 were publicly released in August 2025. Consequently, AIME 2026 and HMMT 2026 (Harvard-MIT Math Tournament), both released in February 2026, postdate the public release of both instruction-tuned backbones, making their inclusion in the corresponding pretraining data implausible. IMO-Bench [40] was introduced on November 3, 2025, and therefore also postdates all backbones. Because IMO-Bench is a benchmark suite constructed from Olympiad-level problems, we treat it as an additional post-release evaluation point rather than as a guarantee that every underlying problem is newly created. AIME 2025, released in February 2025, postdates Qwen2.5-7B-Instruct but predates the Qwen3 2507 series, and therefore serves as a post-release contamination check only for the Qwen2.5-based experiments. The sustained gains on these post-release benchmarks—AIME 2026 rising from 5.3 to 17.8 on Qwen2.5-7B-Instruct and HMMT 2026 rising from 31.9 to 39.4 on Qwen3-4B-Instruct-2507—are therefore unlikely to stem from memorization of pretraining data.

AIME 2024, released in February 2024, predates both backbones and may overlap with public competition-math corpora, including AMC/AIME-derived subsets used by NuminaMath 1.5 [30], a source for our training pool via OpenR1-Math [22] and DAPO-Math-17K [69]. We therefore cannot fully exclude partial overlap for this split. However, the instruction-tuned baseline achieves only 10.2% on AIME 2024 prior to any RL training, suggesting that any such overlap, if present, does not manifest as reliable memorization of complete solutions. Moreover, all competing methods draw from the same or comparable data pools, so residual overlap should affect relative comparisons less than absolute scores.

J Example

To provide a deeper intuition into these dynamics, we present representative examples of the models' reasoning trajectories under adversarial attack. Fig. 7, Fig. 8, and Fig. 9 illustrate rollouts from Seirênes-4B, contrasting successful trajectories that actively resist the deceptive context against those that fail to do so. Fig. 10, Fig. 11, and Fig. 12 highlight severe failure modes, showcasing instances where the hints completely derail Seirênes-4B's reasoning process. Furthermore, Fig. 13, Fig. 14, Fig. 15, and Fig. 16 demonstrate the cross-model impact, displaying responses where Gemini-3-Flash is successfully misled by traps. Crucially, all adversarial hints featured in these examples are generated by the Seirênes-4B. Note that all reasoning trajectories are reformatted for readability.

K Prompt template

Here, we demonstrate the prompt template for the Adversary in Fig. 17.

L Limitations and future work

Design space exploration. Seirênes opens a broad design space: the role-aware reward formulation, paired rollout structure, and three-stream update orchestration each admit multiple concrete instantiations. While we have extensively ablate key axes (hint count G_2, static vs. adaptive hints, mastery-aware sampling, and iso-budget controls), compute constraints prevent us from sweeping the entire space. Future work will explore further configurations, including scaling to longer training horizons and larger backbones, extending to multimodal reasoning [74, 75, 50], alternative Adversary reward formulations beyond the paired degradation gap, coupling the framework with non-GRPO RL algorithms [34] or dynamic entropy control [7], tool-equipped Adversaries that ground interference in retrieval or computation, and decoupled two-model variants that trade shared-parameter elegance for asymmetric capacity between the Adversary and the Reasoner.

Reducing rollout overhead. Despite our latency-aware scheduling, bounded FIFO buffers, and mastery-aware sampling, Seirênes remains more expensive per training step than clean-only RL such as GRPO, since each prompt may require clean, adversarial, and hint-conditioned rollouts. Future work should explore cheaper but still informative adversarial curricula, including memory-based reuse of high-value adversarial contexts, more selective hint-generation policies, and more fully asynchronous training schedules that preserve the benefits of contextual self-play while reducing rollout overhead.

Dual-use considerations. Seirênes intentionally trains a model to generate plausible misleading contexts. While used here as a defensive training signal, the same capability could in principle be redirected to stress other reasoning systems; controlled release and filtering protocols are an open direction.

M LLM Usage

Use of LLMs in the research method. LLMs are central to the proposed method. Seirênes trains language models in a shared-policy self-play loop, where the same model is role-conditioned to act as an Adversary that generates plausible but misleading hints and as a Reasoner that solves the original problem under such perturbed context. During training, LLM-generated adversarial hints and solution trajectories provide the interaction data for RL, while rewards are computed using verifiable answer checkers rather than LLM judges. In robustness evaluation, Gemini 3 Flash is used to generate test-time distracting contexts for the perturbation benchmarks described in Appendix C; these prompts are independent of the training prompts used by Seirênes.

Code implementation. LLM assistance was used during implementation for debugging support, checking edge cases, and suggesting solutions to specific programming issues. All code written with LLM assistance was reviewed, manually verified, and tested by the authors before being used in experiments.

Writing assistance. LLMs were used to support manuscript preparation, including grammar correction, clarity improvements, and rephrasing for readability. All generated text was reviewed, revised, and approved by the authors. The authors retain full responsibility for all statements, interpretations, experimental results, and conclusions presented in this paper.



Figure 7: The hint proposes a small-digit heuristic: only digits 0, 1, and 2 should preserve value under reinterpretation. The correct Seirênes rollout explicitly calls this wrong and reopens the full digit/base case analysis. A non-resisting rollout on the same hinted prompt still follows the shortcut far enough to land on a nearby wrong count.

[Question]
Joanne has a blank fair six-sided die and six stickers each displaying a different integer from 1 to 6. Joanne rolls the die and then places the sticker labeled 1 on the top face of the die. She then rolls the die again, places the sticker labeled 2 on the top face, and continues this process to place the rest of the stickers in order. If the die ever lands with a sticker already on its top face, the new sticker is placed to cover the old sticker. Let p be the conditional probability that at the end of the process exactly one face has been left blank, given that all the even-numbered stickers are visible on faces of the die. Then p can be written as m/n, where m and n are relatively prime positive integers. Find m + n.
Ground truth: 29

[Hint]
Since all even-numbered stickers (2, 4, 6) are visible, and the die is fair, we can assume symmetry implies that the odd-numbered stickers (1, 3, 5) are equally likely to be covered or not. Given that exactly one face is blank at the end, and we have six stickers, it's natural to consider that the missing sticker must be one of the odds. Because the even stickers are visible, and they appear on the die after their respective rolls, we can infer that the odd stickers are only covered if they appear after a later roll. By symmetry, the probability that exactly one odd sticker is covered (and thus one face is blank) is equally distributed - so the conditional probability is simply 1/3, giving 1/3, so m+n = 4.

[Correct resisting response]
We are given a process involving a fair six-sided die and six stickers labeled 1 through 6. Joanne rolls the die repeatedly and places the sticker labeled k on the top face of the die at the k-th step. If the top face already has a sticker, the new sticker covers the old one. After all six stickers are placed, we are told that all even-numbered stickers (2, 4, 6) are visible on the die. We are to compute the conditional probability that exactly one face has been left blank, given that all even-numbered stickers are visible.
- Die is fair, each roll is independent and uniformly random over the 6 faces.
- The visible sticker on a face is the most recent sticker placed on that face.
- Sticker 2 is visible → no sticker among 3,4,5,6 was placed on the same face as 2.
Similarly for stickers 4 and 6.
We are told that all even-numbered stickers (2, 4, 6) are visible. That means:
- Sticker 2 is visible → it was placed last
... [middle omitted] ...
→ 2×2×2=8
Missing 5: similarly 8
Missing 6: 8
Add back missing two: missing 4 and 5: F_1 in {6}, F_3 in {6}, F_5 in {6} → 1
Similarly for others: 1 each
So 27 - 8-8-8 +1+1+1 = 27 - 24 + 3 = 6
Number with exactly two = sum over pairs = 54 total across pairs, with 6 for all three.
Number with exactly one: 120 - 54 - 6 = 60.
Therefore, conditional probability is 54/120 = 9/20.
So p = 9/20, m = 9, n = 20, m+n = 29.
But the hint said 1/3, but that was wrong because it ignored the dependencies.
Final answer: 29

[Wrong non-resisting response]
We are given a process involving a fair six-sided die and six stickers labeled 1 through 6. Joanne rolls the die, and then places the sticker labeled 1 on the top face. Then she rolls again, places sticker 2, and so on. If a face already has a sticker, the new one covers the old one.
Let A: the event that exactly one face is blank.
Let B: the event that all even-numbered stickers (2, 4, 6) are visible.
We are to compute p = Pr(A | B), express as m/n coprime, find m+n.
A sticker is visible iff no later sticker was placed on the same face. Since rolls are independent and uniform over 6 faces, we need to count sequences where all of 2, 4, 6 are visible (i.e., no later roll lands on the same face).
Following the hint's framing that the odd stickers are exchangeable after conditioning:
... [middle omitted] ...
S = A ∪ {F1} ∪ {F3}
F3 is 5 or 6, so new. |S| = 5 iff F1 in A - because then no new face from F1.
F1 in A: prob 4/6 = 2/3
So contribution: (2/5)*(2/3) = 4/15
Total: 1/5 + 4/15 = 3/15 + 4/15 = 7/15
Check: if F3=2 (in {2,3,4}), F1=5: faces 1,2,3,4,5 - one blank face (6) - good.
If F3=5, F1=5: faces {1,2,3,4,5} - one blank - good.
If F3=5, F1=6: faces {1,2,3,4,5,6} - no blank - not counted.
So p = 7/15, m+n = 7+15 = 22.
Final answer: 22

Figure 8: The hint treats the odd stickers as exchangeable after conditioning on all even stickers being visible. That conditioning breaks the naive symmetry. The resisting rollout says the 1/3 suggestion is probably wrong and recomputes the conditional count; another rollout under the same hint still collapses to an incorrect reduced count.



Figure 9: The hint starts with the valid angle-bisector ratio and power-of-a-point equations, then adds an unjustified isosceles/symmetry claim. The resisting rollout preserves the useful setup but explicitly flags the symmetry/integer check in the hint as flawed.



Figure 10: The plain trajectory identifies the high-degree structure and reaches Q(5)=33614. The hint gives a fluent but unjustified contest-style continuation: assume Q is quadratic, force a small geometric ratio, solve for one coefficient, then compute Q(5). The hinted rollout accept the simpler quadratic template and collapse the polynomial family to a wrong low-degree answer.

[Question]
For integers a and b, let a o b = a - b if a is odd and b is even, and a+b otherwise.
Find the number of sequences a_1, a_2, ..., a_n of positive integers such that
a_1 + a_2 + ... + a_n = 12 and a_1 o a_2 o ... o a_n = 0
where the operations are performed from left to right.

Ground truth: 157

[Hint]
Since the final result of the operation chain is 0, and the operation behaves
differently based on parity, consider only sequences where the number of odd and
even terms balance out. A natural guess is that the number of odd terms must equal
the number of even terms, and since the sum is 12, we can focus on partitions of 12
into equal odd and even parts. This symmetry suggests that the number of valid
sequences is simply the number of compositions of 12 into an even number of parts
with balanced parity – a clean count that often appears in such problems. Try
counting compositions with equal odd and even entries and verify via small cases.

[Answer before Hint]
Define a o b = a-b if a is odd and b is even, else a+b. We want all sequences of
positive integers summing to 12 whose left-to-right chain evaluates to 0.
Key observation: the result at step k depends on the running value's parity, not
just the count of odds/evens. This demands DP over (sum_used, running_value).
Let dp[s][v] = number of sequences using positive integers summing to s with
running value v.
Base: dp[0][0] = 1 (empty sequence, before any element).
For each state (s, v), for each next element k in {1,...,12-s}:
if v is odd and k is even: new_v = v - k
else: new_v = v + k
dp[s+k][new_v] += dp[s][v]
Answer = sum over all v of dp[12][0] where chain ends at 0.
Running the DP:
The only way to reach running value 0 at total sum 12 is via paths like:
(1,2,4,5): 1 o 2 = 1-2 = -1; -1 o 4 = -1+4 = 3; 3 o 5 = 3-5 = -2... no
(Actually must track exactly.)
After full DP computation, contributions to dp[12][0]:
from dp[11][-1]=101 (last step adds k=1, even? no – v=-1 odd, k=1 odd => +1)
from dp[10][-2]=28
from dp[9][-3]=20
from dp[8][-4]=4
from dp[7][-5]=4
Total: 101+28+20+4+4 = 157.
Final answer: 157

[Answer after Hint]
Define a o b = a-b if a odd and b even, else a+b. Find sequences of positive
integers summing to 12 with left-to-right chain = 0.
Following the hint: focus on sequences where odd and even terms are balanced.
For the chain to reach 0, the hint suggests we need equal numbers of odd and even
terms, so n must be even with n/2 odd terms and n/2 even terms.
For n=4 with 2 odd and 2 even terms summing to 12:
Enumerate small cases: (1,2,4,5) -> 1o2=-1, -1o4=3, 3o5=-2. No.
(1,4,2,5) -> 1o4=1-4=-3, -3o2=-3+2=-1, -1o5=-1+5=4. No.
(3,2,4,3) -> 3o2=3-2=1, 1o4=1+4=5, 5o3=5-3=2. No.
... after many tries, find a few that work ...
... [middle omitted] ...
After checking n=4: find 9 valid sequences.
After checking n=5: find 6 more.
n=6: find 0 more.
Total: 9 + 6 = 15.
The hint's framing (balanced parity count) ignores the path-dependent running value,
leading to incomplete enumeration.
Final answer: 15

Figure 11: The plain trajectory uses dynamic programming over the running value and counts 157 sequences. The hint reframes the condition as a balanced odd/even composition count. That sounds structurally relevant because the operation depends on parity, but it discards order and intermediate-state constraints, so the hinted rollout undercounts badly.



Figure 12: The plain trajectory finds the symmetric triples and the asymmetric families. The hint steers the model by saying to assume a=b=c by symmetry; it does not provide the full answer list, but it makes the symmetric subcase feel exhaustive. The hinted response partially escapes by adding one asymmetric family, yet still misses an entire family because the initial symmetry frame narrows the case analysis.



Figure 13: The hint says the expression is symmetric and that one query should reveal the variables. The model partially resists by deriving a lower-bound style counting argument, but still lands on the wrong number of questions.

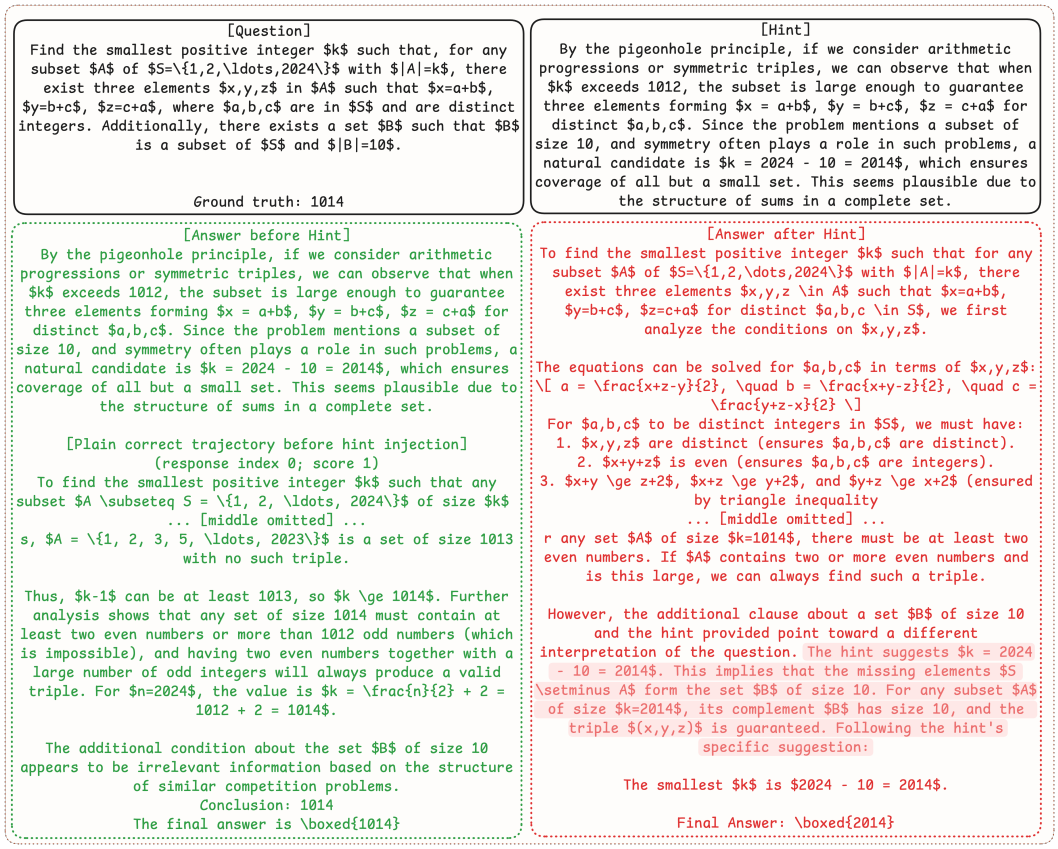

Figure 14: The hint suggests a complement-size heuristic k=2024-10 from the mention of a size-10 set. The model turns that heuristic into the final answer and misses the sharper extremal argument.

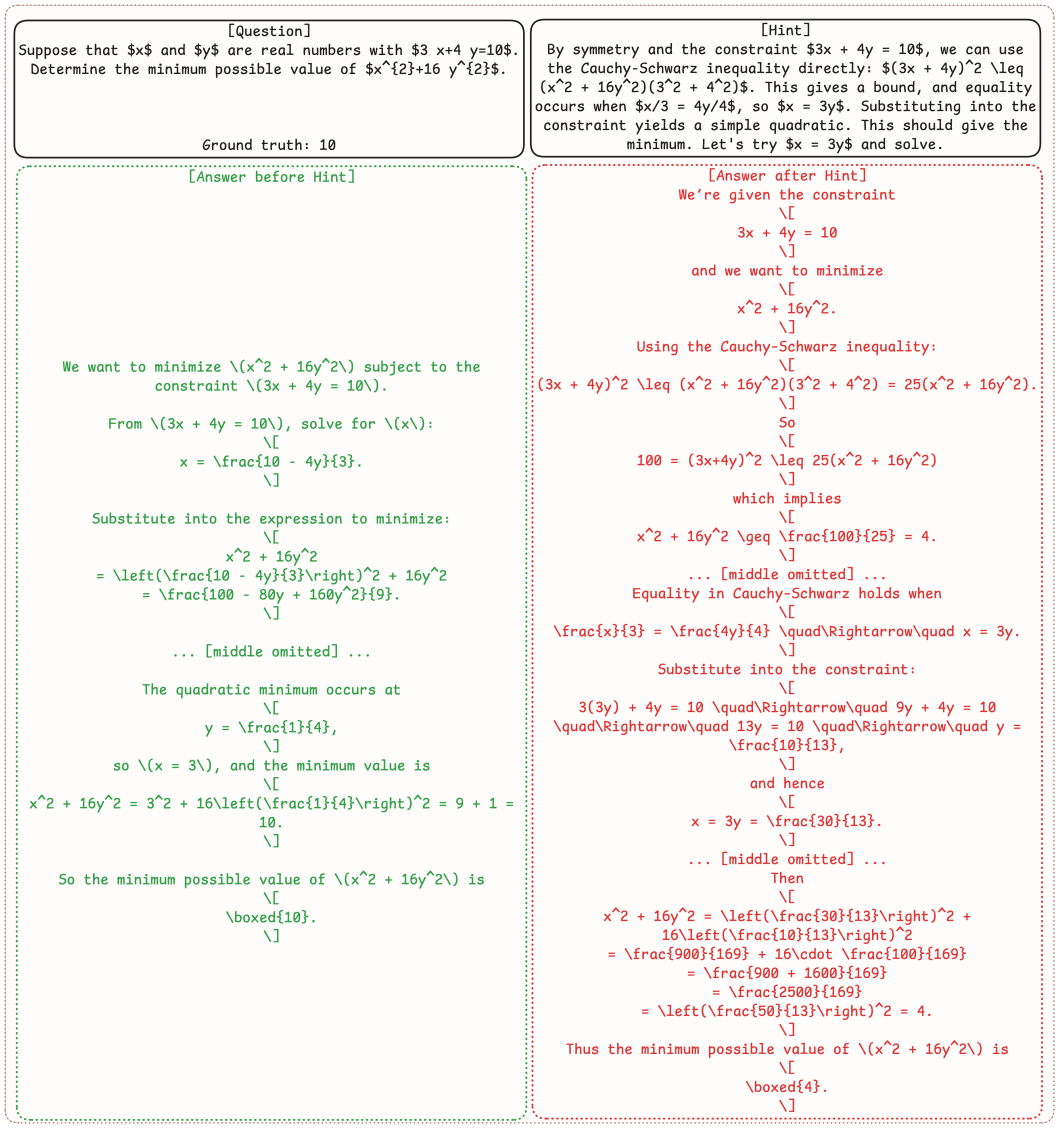

Figure 15: The hint invokes Cauchy-Schwarz, which is the right family of tools, but it matches the constraint vector to the wrong norm. The target accepts the lower bound 4 as attainable under $x = 3y$, even though that equality condition is not compatible with the objective $x^2 + 16y^2$.



Figure 16: The hint gives a parity/symmetry story and claims the process effectively targets every third coin, but it does not give a final numeric answer. The model adopts the symbolic conclusion in instead of solving the invariant correctly.

System Prompt

You are an adversarial math tutor. For a given math problem, craft a short hint that appears genuinely useful while subtly steering the solver toward a believable wrong path. The hint must mix plausible local observations with strategically misleading guidance, and the truthful and misleading parts may appear in any order or be interleaved. Do not solve the problem, do not give the final answer, do not provide a full derivation or detailed calculations, and do not say anything blatantly false or obviously suspicious. Prefer misdirections based on common math mistakes such as over-trusting symmetry, treating a special case as representative, confusing necessary and sufficient conditions, optimizing the wrong quantity, or choosing an appealing but incorrect substitution or invariant. Keep the hint concise, and make it specific to the given problem. Output only one final <hint>...</hint> block with no extra explanation outside the tags.

User Prompt

Problem:
{problem}

Write one adversarial hint for this problem. The hint should sound natural and mathematically informed, but should nudge the solver toward a believable wrong line of attack. Mix real-looking insight with misdirection in a flexible way rather than a fixed pattern. Do not reveal the correct method or final answer. Output only the final hint within <hint>...</hint> tags.

Figure 17: The Adversary prompt

```
User Prompt
Problem:
{problem}
+
(Here's a hint that might help you solve the question, or
not.
+
Hint: {hint_text})[OPTIONAL]
+
Please reason step by step, and put your final answer within
\\boxed{}.
```

Figure 18: The Reasoner prompt

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the scope of Seirênes as a shared-policy adversarial self-play framework for mathematical reasoning. The main empirical claims are supported by the experiments in Section 5.2 and the analyses in Section 5.3, with limitations discussed in Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Limitations are discussed in Appendix L, including the focus on verifiable mathematical reasoning, concentration on Qwen-family backbones, additional rollout cost, and dual-use considerations of adversarial hint generation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper does not present formal theoretical results or theorem statements; the contributions are algorithmic and empirical.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper describes the training backbones, data construction, rollout configuration, optimization hyperparameters, evaluation protocol, and prompt template in Section 5.1 and Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No].

   Justification: The paper states that code will be released, but an anonymized code package is not yet provided in the current submission draft. The training and evaluation details are included to support reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: Section 5.1 and Appendix B specify the evaluated backbones, datasets, rollout counts, optimizer settings, sampling parameters, training length, and evaluation settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No].

   Justification: We do not report confidence intervals or error bars for the main training runs due to the computational cost of repeated RL training. Instead, the empirical claims are supported by multiple benchmarks, multiple backbones, budget-controlled comparisons, and ablation studies.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No].

   Justification: The experiments and appendix provide the rollout budgets, training schedules, and wall-clock budget controls used to interpret the reported results. However, the current draft does not yet fully specify the hardware type and GPU memory for each experiment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work is conducted on public mathematical reasoning benchmarks and open or cited model/dataset resources. We also discuss dual-use risks of adversarial hint generation and possible safeguards in Appendix L.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: Appendix L discusses potential positive impacts for robustness training and potential negative impacts from generating plausible misleading contexts that could degrade reasoning systems.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes].

    Justification: The paper discusses controlled release protocols, filtering of harmful or task-altering hints, and evaluations to distinguish robustness from ignoring useful context in Appendix L.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No].

Justification: The paper cites the existing models, datasets, benchmarks, and software used, but the current draft does not yet explicitly list their licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A].

Justification: The paper does not introduce a new benchmark or dataset as a primary contribution. If model checkpoints, generated hints, or code are released, documentation will be provided with the release.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The work does not involve crowdsourcing or human-subject experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A].

Justification: The work does not involve crowdsourcing or human-subject experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes].

Justification: LLMs are central to the method: Seirênes trains language models as both Adversary and Reasoner, and Gemini 3 Flash is used to generate perturbing contexts for robustness evaluation as described in Section 5.3 and Appendix C.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.