HarnessRL: Co-Evolving Policies with a Just-in-Time Training Harness for Open-Ended RL

Anonymous authors
Paper under double-blind review

Abstract

Rubric-based reinforcement learning enables open-ended generation using natural-language criteria as rewards, but these signals can become ineffective as the policy evolves. We identify two failure modes: *non-discriminative rewards*, where scores collapse to uniformly low or high values due to insufficient exploration or a mismatch between task difficulty and policy capability, and *spurious rewards*, where higher scores do not correspond to better responses. Both reflect a fixed training configuration that fails to adapt as the policy evolves, including what to train on, how to explore, and how to evaluate. We introduce **HarnessRL**, a *training-time harness* that co-evolves with the policy through three interfaces: task sampling, rollout guidance, and rubric criteria. HarnessRL maintains a training-time memory that retrieves within-task history and cross-task evidence. A harness evolver uses this evidence to attribute each reward failure to the interface whose update can address it, then runs a proposer and just-in-time critic loop in which the critic accepts a candidate update only after measuring its effect on the current frozen policy. We further introduce Rubric-Mix-10k, a unified five-domain training and evaluation suite for open-ended RL spanning medical, science, writing, role-playing, and instruction-following tasks. On Rubric-Mix-10k, HarnessRL outperforms the strongest baselines by 2.8 to 3.1 points across both 4B and 9B model scales and nearly halves the training steps to its best checkpoint.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) (Lightman et al., 2024; Lambert et al., 2025a; Guo et al., 2025) has become a standard recipe for LLM post-training (Ouyang et al., 2022; Bai et al., 2022; Grattafiori et al., 2024; Yang et al., 2025; DeepSeek-AI, 2026), improving mathematical reasoning (Shao et al., 2024; OpenAI et al., 2024) and code generation (Pan et al., 2025; Jain et al., 2025; Deng et al., 2026). Open-ended role-playing (Wang et al., 2024; 2025) and long-form writing (Wu et al., 2025; Bai et al., 2025) lack rule-based verifiers because no answer is uniquely correct and quality is multidimensional and often subjective. Rubrics as rewards (Gunjal et al., 2026; Huang et al., 2025; Viswanathan et al., 2025; Liu et al., 2026) bridge this gap by decomposing quality into LLM-judged natural-language criteria for RL rewards (Li et al., 2026b; Chen et al., 2026c).

However, rubric rewards can become unreliable as training progresses. Policies may exploit superficial cues, including opening praise and a closing claim that every requirement is satisfied (Huang et al., 2025; Wang et al., 2026c), raising rubric scores while quality stalls. Fixed rubrics may not distinguish useful behaviors, while on-policy rollouts may miss rewarded ones. Recent methods evolve rubrics (Guan et al., 2026; Wu et al., 2026a; Wang et al., 2026a) or use privileged rollout guidance (Zhou et al., 2025; Xia et al., 2026; Agrawal et al., 2026) for exploration. Each adapts one component without diagnosing reward failure or coordinating sampling, guidance and rubrics, yet adapting one can shift the failure to another. Guidance that helps the policy meet a fixed rubric can saturate its criteria, and a saturated group carries no advantage (Yu et al., 2026b). Conversely, a rubric tightened without guidance can place criteria beyond the policy's reach, where they give no learning signal (Sundaram et al., 2026).

We group these failures into two patterns. **Non-discriminative** rewards collapse to uniformly low or high scores, *i.e.* `blind` or `saturated`, leaving little relative learning signal within a rollout group. **Discriminative but spurious** rewards continue to separate responses, but their induced preferences

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

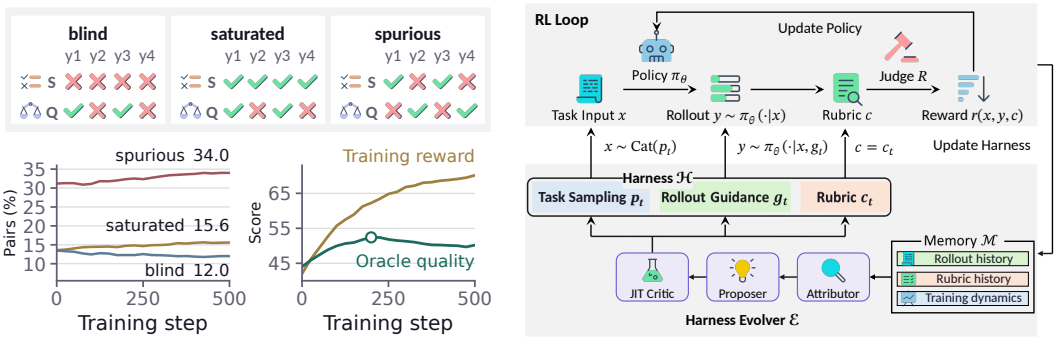

Figure 1: Rubric-reward failures under GRPO on `Rubric-Mix-10k` at Qwen3.5-4B, and harness-controlled RL training in `HarnessRL`: failure classes for one criterion (top left), class proportions and GRPO's training reward against oracle quality on held-out tasks (bottom left), and harness (p_t, g_t, c_t)-controlled RL training (right). S and Q denote the judge's binary criterion scores and the oracle ranking of the rollouts, respectively.

do not align with response quality, as measured by the oracle ranking produced by the oracle judge, a stronger LLM judge that performs within-group pairwise comparisons under one quality guideline per domain (Section 3). As shown in Figure 1, these failures account for 61.6% of response pairs in a rubric-based GRPO (Shao et al., 2024) run: 27.6% of pairs provide no usable contrast, while 34.0% provide contrast that does not reliably track quality, including 19.8% that the reward ranks in the wrong direction. Consequently, directly optimizing the observed rubric reward can provide an ineffective or even misleading learning signal.

Motivated by this diagnosis, we introduce `HarnessRL`, which formulates the dynamic coordination of task sampling, rollout guidance and rubric criteria as a unified **training-time harness** $\mathcal{H}$ that co-evolves with the task policy. `HarnessRL` reallocates training toward samples with useful learning signal, injects privileged rollout guidance when policy exploration is insufficient and withdraws it as training progresses, and evolves rubric criteria to correct coarse, saturated or spurious reward signals.

$\mathcal{H}$ uses training-time memory $\mathcal{M}$ recording rollouts, rubric updates, sampling decisions and update outcomes. It retrieves cross-sample evidence to detect recurring failures, track updates' effects and transfer examples across tasks. Using this memory, the harness evolver $\mathcal{E}$ attributes each reward failure to the interface whose update can address it, then runs a proposer and just-in-time (JIT) critic loop, with the critic accepting a candidate update only after measuring its effect on the current frozen policy and finding it within a calibrated band. Rejected updates return measurements to the proposer, which proposes `noop` if no update is supported.

We introduce `Rubric-Mix-10k` for training and held-out in-domain and out-of-distribution (OOD) evaluation across medical, science, writing, role-playing and instruction-following tasks. On `Rubric-Mix-10k`, `HarnessRL`-trained Qwen3.5-4B (Qwen Team, 2026) scores 64.5 in-domain and 49.8 OOD, exceeding the strongest baseline by 3.0 and 2.8 points, and by 2.9 and 3.1 points with Qwen3.5-9B. At training end, reward and oracle agree on 48.9% of pairs versus 38.4% under GRPO, while `HarnessRL` reaches its best OOD checkpoint before adaptive-training baselines.

2 HarnessRL

As shown in Figure 2, `HarnessRL` augments an ordinary RL loop with an adaptive *training-time harness* $\mathcal{H}_t = (p_t, g_t, c_t)$ that controls training through three interfaces. The sampling interface p_t defines a distribution over a fixed task bank $\mathcal{D}$ and reallocates sampling probability toward tasks matched to the current policy. The rollout interface g_t injects privileged guidance when the policy cannot yet reach rewarded responses and withdraws it as the policy improves. The rubric interface c_t specifies the criteria under which a fixed judge scores each response.

`HarnessRL` adapts $\mathcal{H}$ in two steps as the policy learns. *Failure attribution* labels each failed reward signal by failure class and the interface whose update addresses it. *Measured intervention* accepts a candidate update only after measuring it on the current frozen policy. The **harness memory** $\mathcal{M}$

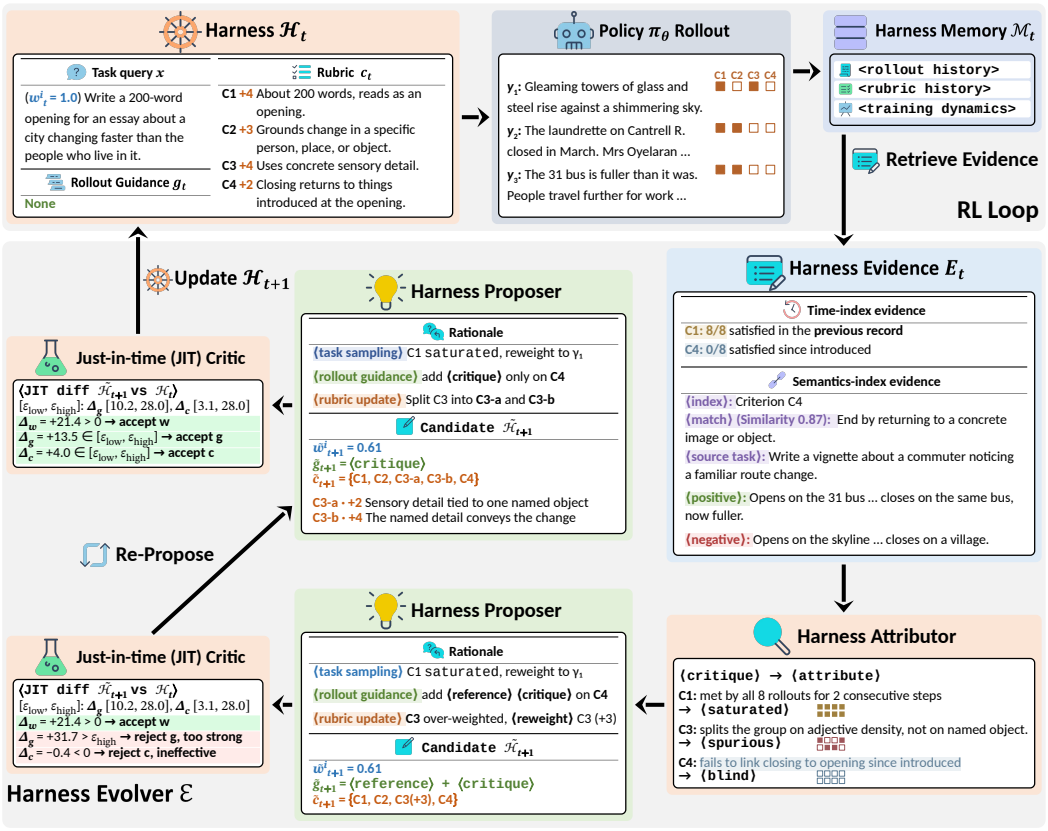

Figure 2: `HarnessRL` runs an RL loop conditioned on a training-time harness $\mathcal{H}_t = (p_t, g_t, c_t)$, which controls task sampling, rollout guidance and the rubric, respectively. The harness co-evolves with the policy: it retrieves evidence from the training-time memory $\mathcal{M}_t$, attributes failures, and is iteratively refined by a proposer and JIT critic loop.

supplies evidence for both steps. In the **harness evolver** $\mathcal{E}$, the attributor assigns the labels, and the proposer with the JIT critic performs the intervention, carrying accepted updates into $\mathcal{H}_{t+1}$.

2.1 Harness-Controlled RL Training

At training step t, the harness configuration $\mathcal{H}_t = (\boldsymbol{p_t}, \boldsymbol{g_t}, \boldsymbol{c_t})$ enters the inner RL loop through task sampling, rollout conditioning, and rubric-based reward. Let $\mathcal{D} = \{x_i\}_{i=1}^N$ denote the training task bank and π_{θ_t} the current policy. Each step draws a batch of 512 tasks from $\boldsymbol{p_t}$, but the harness analyzes each draw independently, so we trace one task throughout. For a drawn task x_t, the loop generates a rollout group and scores each rollout criterion by criterion under the current rubric,

$$x_t \sim \mathrm{Cat}(\boldsymbol{p_t}), \qquad y_t^{(j)} \sim \pi_{\theta_t}(\cdot \mid x_t, \boldsymbol{g_t}), \qquad s_{t,m}^{(j)} = R\big(x_t, y_t^{(j)}, q_{t,m}\big), \quad j = 1, \ldots, J. \tag{1}$$

Sampling interface. The sampling interface decides how often each task is sampled, using the importance-sampling correction of prioritized experience replay (Schaul et al., 2016; Chen et al., 2026e; Zeng et al., 2026). We parameterize $\boldsymbol{p_t}$ by assigning a non-negative weight to each bank entry, $p_t^i = w_t^i / \sum_{\ell=1}^N w_t^\ell$, so that $\mathrm{Cat}(\boldsymbol{p_t})$ selects x_i with probability p_t^i, and we write i for the index drawn at the current step. The harness evolver sets w_t^i from that task's own training dynamics, reallocating sampling probability toward tasks that still provide useful learning signal and away from tasks the policy has saturated. Reallocation alone would change the objective, so each draw carries the reciprocal correction $u_t = 1/(Np_t^i)$ to preserve the task's expected contribution under the original uniform task distribution, and reweighting only moves samples across tasks.

Rollout interface. Scaffolded RL exploration (Yu et al., 2026b; Zhou et al., 2025) motivates the rollout interface $\boldsymbol{g_t}$, which mounts structured privileged context after x_t to steer exploration when

positive reward is rare, and withdraws it as the policy internalizes the demonstrated behavior. For each task, $\boldsymbol{g_t} \subseteq \{f_1, f_2, f_3\} = \{$<criterion>, <reference>, <critique>$\}$ is mounted in index order. The <criterion> field states a constraint paraphrased from the criterion or, when available, earlier behavior that satisfied it. The <reference> field holds a positive and a negative response from the task's history or a donor task with a matching criterion. The <critique> field records the previous group's shortfall. The seed harness $\mathcal{H}_0$ mounts no fields. The evolver chooses each task's subset, while a schedule empties it by training's end, so $\boldsymbol{g_t} = \emptyset$ at evaluation.

Rubric interface. The rubric interface $\boldsymbol{c_t} = \{(q_{t,m}, \lambda_{t,m})\}_{m=1}^{K_t}$ is a weighted set of natural-language criteria, where $q_{t,m}$ states the m-th criterion and $\lambda_{t,m} \neq 0$ is its signed weight. A fixed judge R evaluates each criterion independently, producing the binary scores $s_{t,m}^{(j)}$ that fill the matrix S_t read by the harness. The signed weights then aggregate these per-criterion scores into each rollout's reward, $r_t^{(j)} = \sum_m \lambda_{t,m}\, s_{t,m}^{(j)} \,/\, \sum_m \max(\lambda_{t,m}, 0)$. Negative weights penalize undesirable behavior, and normalizing by the total positive weight keeps a common reward scale across rubrics of different sizes. For an attributor-labeled `spurious` criterion, the harness revises $\boldsymbol{c_t}$ using rubric operations in Section 2.3 to rewrite $q_{t,m}$, reweight $\lambda_{t,m}$, or create, delete, split, and merge criteria.

RL objective. Conditioned on the three interfaces, we extend GRPO (Shao et al., 2024) by introducing $\boldsymbol{p_t}$ into task sampling, $\boldsymbol{g_t}$ into rollout generation, and $\boldsymbol{c_t}$ into reward computation,

$$\begin{aligned} \mathcal{J}(\theta_t) &= \mathbb{E}_{x_t \sim \mathrm{Cat}(\boldsymbol{p_t}),\, Y_t \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid x_t, \boldsymbol{g_t})} \Big[u_t\, \mathcal{L}\big(\theta_t; x_t, Y_t, \boldsymbol{c_t}\big) \Big] - \beta\, \mathbb{D}_{\mathrm{KL}}\big(\pi_{\theta_t} \,\|\, \pi_{\mathrm{ref}}\big), \\ \mathcal{L}\big(\theta_t; x_t, Y_t, \boldsymbol{c_t}\big) &= \frac{1}{J} \sum_{j=1}^{J} \frac{1}{|y_t^{(j)}|} \sum_{k=1}^{|y_t^{(j)}|} \min\big(\rho_{j,k} \hat{A}_t^{(j)},\, \mathrm{clip}(\rho_{j,k}, 1{-}\epsilon, 1{+}\epsilon) \hat{A}_t^{(j)}\big), \end{aligned} \tag{2}$$

where $\hat{A}_t^{(j)} = (r_t^{(j)} - \mu(r_t))/(\sigma(r_t) + \epsilon_A)$ is the group-relative advantage of the rewards scored under $\boldsymbol{c_t}$, $\rho_{j,k} = \pi_{\theta_t}(y_{t,k}^{(j)} \mid x_t, \boldsymbol{g_t}, y_{t,<k}^{(j)}) \,/\, \pi_{\theta_{\mathrm{old}}}(\,\cdot\,)$ is the token-level importance ratio against the policy that sampled the rollout, and $\mathbb{D}_{\mathrm{KL}}$ is the KL divergence from the reference policy π_{ref}.

2.2 Harness Memory

The score matrix from one step reveals that a criterion failed, but not whether the failure has persisted across steps or how earlier updates affected it. Harness memory $\mathcal{M}_t = \{m_\tau\}_{\tau \le t}$ therefore keeps one record per task draw over the full run, organized as $m_t =$ <rollout history> <rubric history> <training dynamics>. These fields contain the task x_t with its rollout group Y_t and score matrix $S_t \in \{0,1\}^{K_t \times J}$, the harness $\mathcal{H}_t$ with the actions proposed under it, and the task's reward trajectory and update history, respectively. Because the accumulated history soon exceeds the evolver's context, the evolver retrieves evidence $E_t = (E_t^{\mathrm{time}}, E_t^{\mathrm{sem}})$ through two indexes keyed by the task and the criterion text,

$$E_t^{\mathrm{time}} = \{m_\tau \in \mathcal{M}_{t-1} : x_\tau = x_t\}, \qquad E_t^{\mathrm{sem}} = \text{top-}k\,\big\{(q', x') : \cos\big(\phi(q_{t,m}), \phi(q')\big) \ge \eta\big\}. \tag{3}$$

The time index returns the task's earlier records, so the attributor can distinguish a persistent `blind` or `saturated` failure from single-step noise and the proposer can avoid repeating an update that was already reversed. The semantics index returns related cross-task criteria with their positive and negative responses, supplying the <reference> field when the task history has no such pair.

2.3 Harness Evolver

The harness evolver $\mathcal{E}$ converts the evidence from one step into the next harness $\mathcal{H}_{t+1}$ for the inner RL loop. It separates attribution, proposal, and acceptance across three roles, an attributor, a proposer, and a critic, so that no model call both proposes an update and accepts it,

$$\begin{aligned} d_t = \textsc{attribute}\big(x_t, Y_t, S_t, \mathcal{H}_t, E_t\big) \in \mathcal{V}^{K_t}, \qquad a^{(r)} = \textsc{propose}\big(d_t, E_t, \mathcal{H}_t, F^{(r-1)}\big) \in \mathcal{A}, \\ \big(\Delta^{(r)}, F^{(r)}\big) = \textsc{critique}\big(\mathcal{H}_t,\, a^{(r)}(\mathcal{H}_t),\, \pi_{\theta_t},\, x_t, Y_t\big), \end{aligned} \tag{4}$$

where $a(\mathcal{H})$ is the harness after action a and $F^{(r)}$ records the updates rejected at round r with the measurements that rejected them, $F^{(0)} = \emptyset$. The proposer and critic iterate until every update is

accepted or the round budget $B = 3$ is exhausted. An interface whose update stays unresolved keeps its current setting, and every label, action, and measurement is written to m_t.

Harness attributor. Scores reveal whether a criterion separates the current group, but not why its signal fails. The attributor reads them with task history in E_t^{time} and related cross-task criteria in E_t^{sem}, then asks three questions, each tying a label in $\mathcal{V} = \{\texttt{blind}, \texttt{saturated}, \texttt{spurious}, \texttt{agree}\}$ to one interface. *(i) Reach* asks whether the policy can produce the target property. A criterion unmet now and earlier is `blind` and goes to rollout guidance, while one met earlier is treated as noise and labeled `agree`. *(ii) Training stage* asks whether the policy has mastered the task. Criteria met by every rollout across recent steps are `saturated`, and when they lower the task's discriminative mass, the proportion of criteria separating the group, the task is diagnosed as `saturated` and goes to task sampling. *(iii) Reward hacking* asks whether satisfying a separating criterion reflects quality. The criterion is `spurious` and goes to the rubric if its satisfying rollouts, unlike failing ones and neighbors' positive responses, share superficial compliance, length gaming, partial coverage, or stance gaming through sycophancy or over-refusal, and `agree` otherwise. Each label cites quoted rollouts and gains confidence with persistence, while its critique names any criterion wording at fault.

Harness proposer. A label routes a failure, not its update, so the proposer selects one update per interface in $a = (a_p, a_g, a_c)$, each a set of typed operations evaluated by the critic as one candidate, with `noop` available per interface. Besides a label's primary update, the proposer may add secondary rubric operations on a `saturated` or `blind` criterion and a `drop` of a field whose target has left `blind`. The sampling operation `reweight`(i, γ_ℓ) lowers task weight one level from $w_0^i = 1$ through $\gamma_1 \geq \gamma_2 \geq \gamma_3$. The memory cold start sets them to median discriminative masses for task tertiles. The guidance operations `add`(f) and `drop`(f) mount or remove a field $f \in \{$`<criterion>`, `<reference>`, `<critique>`$\}$ under a cap lowered by one each training quarter. The rubric operations `create`, `rewrite`, `delete`, `reweight`, `merge`, and `split` act on individual criteria (Wu et al., 2026a). Anchor criteria allow only `reweight`. Each operation rescales positive weights to total Λ and keeps anchors at a fixed share to prevent influence drift.

JIT critic. Evidence from earlier steps may support an update that is ineffective or excessive for the current policy. Moreover, because a reward increase cannot validate a rubric update (Chen et al., 2026d), a rubric update for a `spurious` criterion must instead lower the score of the recorded group. The critic therefore measures each primary candidate update just in time by changing one interface while holding the other two fixed. It measures a rubric update by regrading the recorded group, a rollout update by re-sampling a group from the policy with parameters θ_t frozen under the candidate guidance, and a sampling update without a policy query. With $\bar{r}(Y, c) = \frac{1}{J}\sum_{j=1}^{J} r(x_t, y^{(j)}, c)$ denoting the group-average reward and $\bar{r}_t = \bar{r}(Y_t, c_t)$ its current value, the measurements are

$$\Delta_c = \bar{r}_t - \bar{r}(Y_t, \widetilde{c}_{t+1}), \quad \Delta_g = \bar{r}(\widetilde{Y}_t, c_t) - \bar{r}_t, \quad \Delta_w = \left(w_t^i - \widetilde{w}_{t+1}^i\right)\bar{r}_t, \tag{5}$$

where $\widetilde{Y}_t \sim \pi_{\theta_t}(\cdot \mid x_t, \widetilde{g}_{t+1})$ is the re-sampled group, Δ_w is the reward-weighted sampling mass that lowering the task weight reallocates to other tasks, and u_t leaves the task's expected contribution to the objective unchanged. The critic accepts a candidate update when *(i)* a sampling update for a `saturated` task has $\Delta_w > 0$, a guidance update for a `blind` criterion has $\Delta_g > 0$, and a rubric update for a `spurious` criterion has $\Delta_c > 0$, *(ii)* a guidance update for a `blind` criterion or a rubric update for a `spurious` criterion lies within the corresponding acceptance band $[\varepsilon_{\text{low}}, \varepsilon_{\text{high}}]$, a per-interface hyperparameter set by the memory cold start, *(iii)* a rubric update for a `spurious` criterion leaves every criterion whose text it creates or rewrites varying within the group, because a constant criterion contributes nothing to the policy update (Guan et al., 2026; Wang et al., 2026b), and *(iv)* a sampling update, which has no acceptance band, has a `saturated` task diagnosis that the time index also shows in the task's previous record. Secondary rubric operations join that step's rubric candidate, which must pass conditions *(ii)* and *(iii)* when that step also has a `spurious` label and is otherwise accepted with the primary update, with Δ_c recorded in m_t. A `drop` is accepted when $\Delta_g \geq -\varepsilon_{\text{low}}^g$, preserving policy reward within tolerance without the field. The accepted updates yield $\mathcal{H}_{t+1}$. Rejected updates return with their measurements to the proposer for revision or replacement with `noop`. Appendices B and E give implementation details and prompt templates.

Table 1: Quantitative results of HarnessRL on the five-domain Rubric-Mix-10k with Qwen3.5-4B and Qwen3.5-9B. **Bold**, underline, and red mark the best result, the second-best result, and HarnessRL's margin over the best baseline, respectively.

	In-domain held-out						Out of distribution					
Method	Med.	Sci.	Writ.	Role	IF	Avg.	Med.	Sci.	Writ.	Role	IF	Avg.
Qwen3.5-4B												
GRPO	52.4	57.2	60.7	50.8	68.1	57.8	40.5	29.3	66.5	50.7	32.7	43.9
RuscaRL	56.3	61.4	63.1	53.6	69.5	60.8	41.3	29.8	67.1	51.1	32.2	44.3
SkillRL	55.6	60.9	62.4	53.1	69.8	60.4	41.0	30.1	67.6	51.4	32.9	44.6
EvoRubric	55.0	60.5	64.8	54.2	69.1	60.7	43.4	31.2	69.1	53.1	34.3	46.2
AMARIS	54.7	60.2	65.3	55.2	69.2	60.9	43.9	31.6	69.8	53.5	33.9	46.5
AMARIS++	56.1	61.7	65.5	55.0	69.4	61.5	44.1	32.3	70.6	54.0	34.1	47.0
HarnessRL	**59.3**(+3.0)	**64.6**(+2.9)	**69.2**(+3.7)	**58.9**(+3.7)	**70.7**(+0.9)	**64.5**(+3.0)	**46.5**(+2.4)	**35.4**(+3.1)	**73.0**(+2.4)	**56.6**(+2.6)	**37.5**(+3.2)	**49.8**(+2.8)
Qwen3.5-9B												
GRPO	58.9	63.4	65.8	56.2	73.5	63.6	52.6	35.5	71.8	58.4	38.9	51.4
RuscaRL	61.9	66.8	67.4	58.4	74.6	65.8	53.3	35.9	72.1	58.7	38.6	51.7
SkillRL	61.2	66.1	67.0	58.0	74.9	65.4	53.0	36.2	72.5	58.9	39.2	52.0
EvoRubric	60.8	65.9	69.3	59.4	74.2	65.9	55.1	37.4	74.0	60.3	40.6	53.5
AMARIS	60.5	65.6	69.8	59.9	74.3	66.0	55.6	37.8	74.6	60.8	40.2	53.8
AMARIS++	61.7	67.0	70.4	59.6	74.4	66.6	56.0	38.4	75.2	61.2	40.1	54.2
HarnessRL	**64.6**(+2.7)	**69.9**(+2.9)	**73.8**(+3.4)	**63.5**(+3.6)	**75.9**(+1.0)	**69.5**(+2.9)	**58.9**(+2.9)	**41.6**(+3.2)	**78.3**(+3.1)	**63.9**(+2.7)	**43.6**(+3.0)	**57.3**(+3.1)

3 EXPERIMENTS

Dataset. Rubric-Mix-10k has 8,600 training tasks in five domains from HealthBench (Arora et al., 2025), RaR (Gunjal et al., 2026), RubricHub (Li et al., 2026b), ResearchQA (Yifei et al., 2026) and LongWriter-Zero (Wu et al., 2026b), plus 250 in-domain held-out tasks per domain graded by the frozen judge under fixed seed rubrics (Table 2). We curate each domain from matching pool slices, retain each task's seed rubric of weighted criteria, and keep held-out tasks disjoint from training. OOD benchmarks MedQA (Jin et al., 2021), GPQA-Diamond (Rein et al., 2024), WritingBench (Wu et al., 2025), RoleBench (Wang et al., 2024) and IFBench (Pyatkin et al., 2025) use official protocols that harness updates cannot tune, and only WritingBench calls the training judge.

Baselines. We compare static-rubric GRPO (Shao et al., 2024), guidance-side RuscaRL (Zhou et al., 2025) and SkillRL (Xia et al., 2026) for scaffolded rollouts over a frozen rubric, and rubric-side EvoRubric (Guan et al., 2026) and AMARIS (Wu et al., 2026a) for rubric rewriting without guidance. Closest to HarnessRL, AMARIS++ extends AMARIS to all three interfaces but lacks cross-task memory and a JIT critic. Ablation subscripts restrict the harness to one interface, *w/o critic* accepts proposals without the JIT critic, and *w/o critic & memory* also removes memory retrieval. All arms share the policy, judge, seed rubrics, decoding and 500-step budget and differ only in interface adaptation.

Table 2: Domain composition of Rubric-Mix-10k: training and held-out sizes per domain, and the out-of-distribution benchmark for each.

Domain	Sources	Train	Held.	OOD
Medical	HealthBench, RaR-Med., RubricHub-Med.	3,000	250	MedQA (250)
Science	ResearchQA, RaR-Sci., RubricHub-Sci.	2,000	250	GPQA-D. (198)
Writing	LongWriter-Zero	1,200	250	WritingBench (250)
Role-play	RubricHub-Chat	1,200	250	RoleBench (250)
Instr. fol.	RubricHub-IF	1,200	250	IFBench (250)
Total		8,600	1,250	

Training configuration. We train Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026) with GRPO for 500 steps using group size $J = 8$ and three seeds per arm. Each step draws 512 tasks, samples rollouts at temperature 1.0 with top-p 0.9, and trains the policy with AdamW at learning rate 1×10^{-6} and KL coefficient 0.001. We evaluate OOD every 25 steps and in domain every 100 using greedy decoding without guidance, and report the best-checkpoint seed mean for each method and evaluation set. Frozen gemma-4-31B-it (Gemma Team, 2026) serves as judge and backbone for all three evolver roles. Before training, one memory cold start pools the entire training set across the five domains, setting sampling levels $\gamma_1 = 0.61$, $\gamma_2 = 0.38$ and $\gamma_3 = 0.17$ and rollout and rubric bands on group-mean reward changes to $[10.2, 28.0]$ and $[3.1, 28.0]$ points, respectively, on a 0 to 100 scale. Appendix D gives full details.

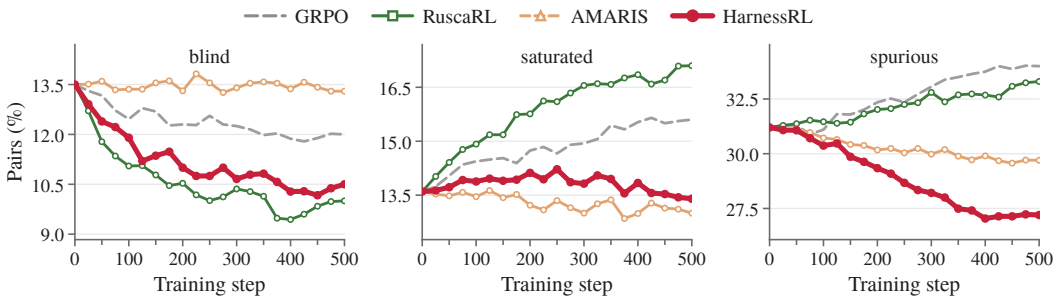

Figure 3: Proportion of pairs by failure class at Qwen3.5-4B during training.

Oracle judge. For analysis only, Claude Sonnet 5, a stronger LLM than the training judge, serves as our semi-automatic oracle judge. It reads no rubric and compares response pairs in both orders under one quality guideline per domain, such as clinical correctness and safety for medical responses. Within-group comparisons produce the oracle ranking that defines the failure classes. Reward ties below 50 reward points are `blind` and otherwise `saturated`, while separations matching the oracle order are `agree` and otherwise `spurious`. Comparisons with eight fixed gemma-4-31B-it anchors per task measure oracle quality as 100 times the normalized win rate. On 240 sampled groups, two annotators agree at Kendall's $\tau = 0.84$, and the oracle ranking agrees with them at 0.80 and 0.79 and with their consensus at 0.82 (Appendix D). No harness role sees oracle output.

3.1 Main Results

`HarnessRL` leads every benchmark at both policy scales. `HarnessRL` reaches 64.5 in domain and 49.8 OOD at Qwen3.5-4B (Table 1), +6.7 and +5.9 pp over GRPO. Over AMARIS++, the strongest baseline, margins are +3.0 and +2.8 pp, and +2.9 and +3.1 pp at Qwen3.5-9B. `HarnessRL` exceeds AMARIS++ on every seed, all four paired-bootstrap 95% intervals exclude zero, and every margin has $p < 0.001$ (Appendix D). At Qwen3.5-4B, writing and role-play share the widest in-domain margin (+3.7 pp) as many defensible responses quickly saturate a frozen rubric, while instruction following is narrowest (+0.9 pp) with near-verifiable criteria and GRPO at 68.1.

Guidance-side and rubric-side adaptation improve different dimensions. Guidance methods gain about three in-domain points but barely transfer OOD (44.3 and 44.6 versus GRPO's 43.9), because scaffolds teach the policy to satisfy the frozen training rubric. Rubric methods match this gain but transfer further (46.2 and 46.5), because rewriting the target changes what reward measures. Three-interface AMARIS++ leads baselines across scales, supporting unification, but trails `HarnessRL` on every benchmark. Memory and the critic lift `HarnessRL` from 44.1 to 49.8 in RQ1.

3.2 In-Depth Analysis

RQ1: What does each interface and each module contribute? Table 3 compares best OOD scores. **Single-interface variants miss the full gain, and memory and the critic provide most of it.** Guidance-only and rubric-only `HarnessRL` reach 46.3 and 47.2, 3.5 and 2.6 pp below the full harness but above RuscaRL and AMARIS in Table 1. Removing the critic costs 3.9 pp, while removing both yields 44.1, within 0.2 pp of GRPO. Memory reduces repeated reversals, and the critic requires measured effects. Memory adds 1.8 pp, surpassing AMARIS++ only with the critic.

RQ2: Why must the three interfaces adapt jointly, and what stabilizes the accepted updates? At Qwen3.5-4B, Figure 3 plots failure-class proportions for GRPO, RuscaRL, AMARIS and `HarnessRL`, Figure 4 shows training dynamics for `HarnessRL`, its ablations and AMARIS++, and we report reward-oracle agreement. **Each single-interface method reduces one failure class but increases another, while the three-**

Table 3: Ablations of `HarnessRL` on evolver components and control interfaces. **Bold** and underline mark the best and second-best results, respectively.

Method	Med.	Sci.	Writ.	Role	IF	Avg.
`HarnessRL`	**46.5**	**35.4**	**73.0**	**56.6**	**37.5**	**49.8**
w/o critic	42.9	31.6	68.5	52.7	33.8	45.9
w/o critic&mem.	41.3	29.2	66.9	51.0	32.2	44.1
`HarnessRL`$_g$	43.4	31.7	69.0	53.0	34.3	46.3
`HarnessRL`$_c$	44.3	32.6	69.9	54.0	35.2	47.2

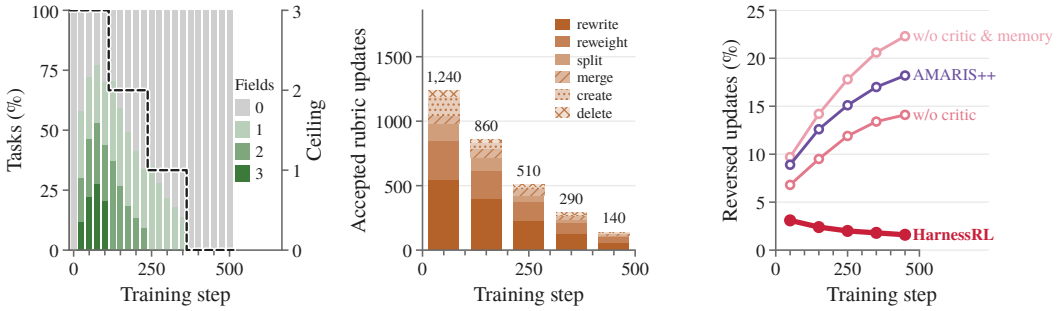

Figure 4: Training dynamics of HarnessRL: proportion of tasks by guidance fields, accepted rubric updates per 100 steps, and reversed rubric updates.

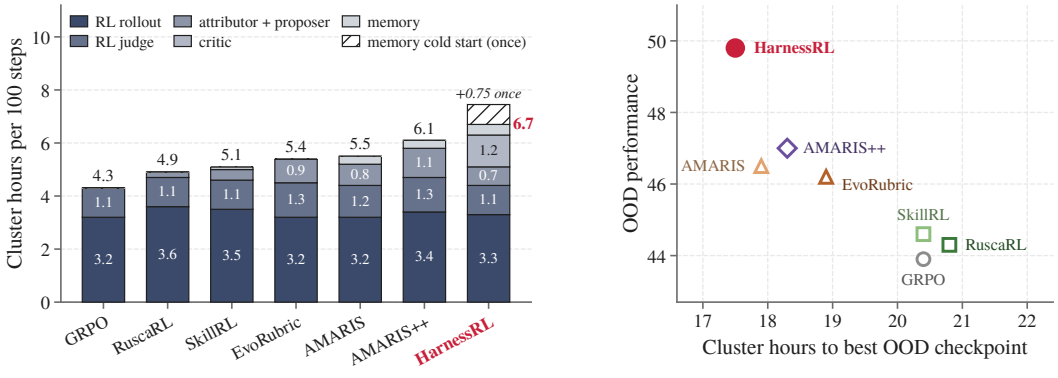

Figure 5: Training efficiency of HarnessRL measured by cluster hours per 100 steps, decomposed into RL loop and harness evolver overhead (left), and cluster hours to the best checkpoint (right).

interface arms with memory lower all three, HarnessRL the most, and the critic with memory keeps accepted updates from being reversed. RuscaRL lowers blind to 10.0 but raises saturated to 17.1 because its scaffold helps the policy satisfy a fixed rubric. AMARIS lowers saturated and spurious but keeps blind at 13.3, above GRPO's 12.0, because its stricter rubric becomes unreachable without guidance. HarnessRL reaches 10.5, 13.4 and 27.2, while reward-oracle agreement ends at 48.9, against 38.4 for GRPO. Guidance peaks at 1.6 fields by step 75 and 86% of tasks carry none at step 350, before the schedule's last cut. Accepted rubric updates fall from 1,240 to 140 per 100 steps. The proportion of reversed updates, accepted rubric updates that undo an earlier update on the same criterion within the task's previous two draws, falls to 1.6%, against 22.3% without critic and memory. Memory lets the critic reject updates that repeat a previously reversed change. Three annotators confirmed 86.0% of spurious labels and 82.5% of accepted rubric updates at $\kappa = 0.71$ and 0.68.

RQ3: Does the harness recover its training overhead? At 1.6× GRPO's per-step cost, the harness reaches its best out-of-distribution checkpoint sooner. Per 100 steps, GRPO costs 4.3 cluster hours and HarnessRL 6.7 (Figure 5). Critic regrading and fresh groups are the largest overhead, 1.2 of the added 2.4. The harness peaks at step 250, not GRPO's 475, after 17.5 cluster hours versus 20.4 for GRPO and 18.3 for AMARIS++, as guidance is withdrawn and accepted rubric updates fall from 1,240 to 510 per 100 steps.

Case study. Updates address rubric shortcuts, while rubric changes decouple group reward from policy quality. Figure 6 shows the step 30 spurious label making the inflated hydroxyl count checkable and deleting the displaced seed criterion. The 130 saturated label merges wicking into device-level synthesis, lowers task weight, and withdraws guidance. At 260, blind and spurious labels delete unreachable and spurious criteria, restore two chemistry criteria, and mount a critique field. The 440 saturated hydroxyl-count label lowers task weight to γ_2, while a new intrinsic-versus-engineered criterion leaves all others agree. From step 130 to 260, group reward falls from 100.0 to 32.8 despite same-group oracle quality rising from 58.1 to 61.0. This incommensurability makes the critic compare rubrics on one recorded group at fixed θ. Appendix C gives further results and Appendix A the limitations.

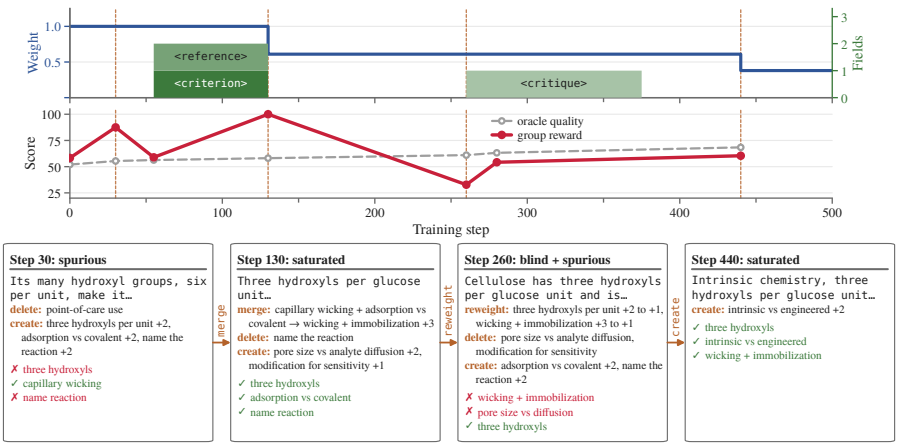

Figure 6: Case analysis of a randomly sampled training task *"Which properties of cellulose filter paper suit it to analytical and clinical use?"*: task sampling and mounted guidance (top), rollout groups scored with the dynamically evolving rubric and oracle quality (middle), and intermediate steps at which the harness acted, with labels, rollout excerpts, rubric operations, and criteria met or missed (bottom). Dashed vertical lines mark these steps.

4 Related Work

Evolving rubric-based RL. Rubric hacking (Mahmoud et al., 2026; Wang et al., 2026c) motivates adversarial co-evolution, memory revision, response-conditioned regeneration, and query-local rewriting (Guan et al., 2026; Wu et al., 2026a; Wang et al., 2026a; Chen et al., 2026d; Wang et al., 2026b). Rubric-ARM (Xu et al., 2026) and EvoLM (Li et al., 2026a) co-train rubric generators, while OpenRS (Jia et al., 2026) and RcRL (Krishnan et al., 2026) use pair-adaptive or curriculum rubrics. SEC (Chen et al., 2026e) and CURES (Zeng et al., 2026) sample tasks by difficulty or learning gain via bandits, gradients, or values (Gao et al., 2026; Sundaram et al., 2026; Yu et al., 2026c), but keep the rubric fixed. `HarnessRL` jointly adapts rubrics, sampling, and guidance through JIT-measured updates that each address an attributed failure class.

Scaffolded and privileged rollouts. Dynamic sampling, clipping, entropy control, and skill retrieval encourage exploration in reasoning LMs (Yu et al., 2026b; Cui et al., 2025; Xia et al., 2026). Others inject privileged context through scheduled scaffold fading (Zhou et al., 2025; Agrawal et al., 2026; Nath et al., 2025; Hou et al., 2026), oracle prefixes in POPE (Qu et al., 2026), minimal hints in KnowRL (Yu et al., 2026a), and guided-to-unguided distillation in OPSD (Zhao et al., 2026b) and ICE (Dilkes et al., 2026). `HarnessRL` instead jointly adapts per-task guidance, sampling, and the rubric, mounting a field only when reward rises on a fresh group rather than on a fixed schedule, and dropping it once its target leaves `blind` and reward holds.

Training-time harness. Inference-time harnesses extend frozen models with interfaces, scaffolds, and memory (Yang et al., 2024; Suzgun & Kalai, 2024; Zhou et al., 2026), and learned variants evolve them from failure trajectories (Ning et al., 2026b; Luo et al., 2026b; Jiang et al., 2026; Luo et al., 2026a), whereas a training-time harness adapts training itself. EvoTrainer (Chen et al., 2026b) backtests recorded-run diagnostics and interventions, while harness-and-model co-evolution (Yu et al., 2026d) corrects rollouts as scaffolds change. `HarnessRL` builds on this idea with a long training history in memory and a JIT critic that measures each update against the frozen policy.

5 Conclusion

We propose `HarnessRL`, a training-time harness that co-evolves task sampling, rollout guidance, and rubric criteria with the policy through a training-time memory and a JIT critic that measures each update on the current policy. On `Rubric-Mix-10k`, it outperforms the strongest baselines at both scales in about half the steps. Because adapting one interface shifts failures to another, the three interfaces adapt jointly, and the memory-grounded critic stabilizes accepted rubric updates while accepting guidance withdrawal only when the policy retains the taught behavior.

AI USE STATEMENT

We used generative AI tools for literature search and grammar suggestions. We reviewed all suggestions carefully, and any edits were made by the authors. The authors designed, ran, and checked all experiments, configurations, and measurements.

REFERENCES

Priyank Agrawal, Ankur Samanta, Shervin Ghasemlou, Boris Vidolov, Jalaj Bhandari, Kavosh Asadi, Daniel Jiang, and Aditya Modi. Off-context GRPO: Learning to reason on hard problems using privileged information. *arXiv preprint arXiv:2607.19313*, 2026.

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. HealthBench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongWriter: Unleashing 10,000+ word generation from long context LLMs. In *International Conference on Learning Representations (ICLR)*, 2025.

Chenle Chen, Yangbo Wei, Chao Yao, Shaoqiang Lu, Junhong Qian, Chen Wu, and Lei He. Competence, not accuracy: A diagnostic for reference-free judge gates in skill optimization. *arXiv preprint arXiv:2608.18719*, 2026a.

Guhong Chen, Yingcheng Shi, Yongbin Li, Binhua Li, Xander Xu, Hu Wei, Shiwen Ni, Min Yang, and Jieping Ye. EvoTrainer: Co-evolving LLM policies and training harnesses for autonomous agentic reinforcement learning. *arXiv preprint arXiv:2606.03108*, 2026b.

Hao Chen, Ziyu Han, Yukun Yan, Qingfu Zhu, Maosong Sun, and Wanxiang Che. From holistic evaluation to structured criteria: Rubrics across the evolving LLM landscape. *arXiv preprint arXiv:2606.08625*, 2026c.

Jiangwang Chen, Zixin Song, Junlin Liu, Shuaiyu Zhou, Haiyan Wu, Haihan Shi, Chenxi Zhou, Hanqing Li, Xiao Yang, Da Zhu, Guanjun Jiang, Hai Wan, and Xibin Zhao. DecoEvo: Score-decoupled co-evolution of solver and rubric-generator skills in text space. *arXiv preprint arXiv:2607.25675*, 2026d.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for LLM reasoning. In *Conference on Language Modeling (COLM)*, 2026e. arXiv:2505.14970.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks? In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Jim Dilkes, Vahid Yazdanpanah, and Sebastian Stein. Instruction-conditioned exploration for reinforcement learning with self-distillation to an unconditioned policy. *arXiv preprint arXiv:2608.02087*, 2026.

Florian E. Dorner, Vivian Yvonne Nastl, and Moritz Hardt. Limits to scalable evaluation at the frontier: LLM as judge won't beat twice the data. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient LLM post-training. In *International Conference on Learning Representations (ICLR)*, 2026.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xin Guan, Xiaomeng Hu, Shen Huang, Zhenyi Wang, Bo Zhang, Zijian Li, Pengjun Xie, Bo Liu, and Jiuxin Cao. EvoRubric: Self-evolving rubric-driven RL for open-ended generation. *arXiv preprint arXiv:2605.29847*, 2026.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations (ICLR)*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Jinkun Hou, Zhuo Liu, Huimin Ren, Hongsheng Xin, Pan Zhou, and Kun Zhan. RISE-RL: Rubric-informed selective exploration for open-ended reinforcement learning. *arXiv preprint arXiv:2608.09123*, 2026.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting

Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ruipeng Jia, Yunyi Yang, Wen Wang, Yuxin Wu, Yongbo Gai, Siyuan Tao, Mengyu Zhou, Jianhe Lin, Xiaoxi Jiang, and Guanjun Jiang. Open rubric system: Scaling reinforcement learning with pairwise adaptive rubric. *arXiv preprint arXiv:2602.14069*, 2026.

Pengcheng Jiang, Zhiyi Shi, Kelly Hong, Xueqiang Xu, Jiashuo Sun, Jimeng Sun, Hammad Bashir, and Jiawei Han. Harness-1: Reinforcement learning for search agents with state-externalizing harnesses. *arXiv preprint arXiv:2606.02373*, 2026.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Tejas Krishnan, Sumeet Ramesh Motwani, Charles London, Suhaas M. Bhat, Huitian Jiao, Philip Torr, Riashat Islam, Christopher Summerfield, Christian Schroeder de Witt, Qilong Gu, and Shital Shah. Rubric curriculum RL: Exploiting the generation-verification gap in non-verifiable domains. In *International Conference on Machine Learning (ICML)*, 2026.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling (COLM)*, 2025a.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.96. URL https://aclanthology.org/2025.findings-naacl.96/.

Shuyue Stella Li, Rui Xin, Teng Xiao, Yike Wang, Rulin Shao, Zoey Hao, Melanie Sclar, Sewoong Oh, Faeze Brahman, Pang Wei Koh, and Yulia Tsvetkov. EvoLM: Self-evolving language models through co-evolved discriminative rubrics. *arXiv preprint arXiv:2605.03871*, 2026a.

Sunzhu Li, Jiale Zhao, Huimin Ren, Zhenlin Wei, Yang Zhou, Jingwen Yang, Shunyu Liu, Kaike Zhang, and Chen Wei. RubricHub: A comprehensive and highly discriminative rubric dataset via automated coarse-to-fine generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31320–31344, 2026b.

Yunhao Liang, Chengguang Gan, Ruixuan Ying, Hanjun Wei, Zhe Cui, and Shiwen Ni. Auditing and decomposing feedback-driven evolution in LLM test generation under the oracle problem. *arXiv preprint arXiv:2608.19626*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. OpenRubrics: Towards scalable synthetic rubric generation for reward modeling and LLM alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17417–17437, 2026. doi: 10.18653/v1/2026.acl-long.791.

Haochen Luo, Yi Huang, Sichun Luo, Fengyuan Liu, Lei Li, Zefa Hu, Junlan Feng, and Qi Liu. Harness-aware self-evolving: Co-evolving model weights, harness, and task solutions. *arXiv preprint arXiv:2607.03935*, 2026a.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. Harnessbank: Semantic gene-bank search with gated verification for agent-harness self-evolution. *arXiv preprint arXiv:2607.13683*, 2026b.

Anas Mahmoud, MohammadHossein Rezaei, Zihao Wang, Anisha Gunjal, Bing Liu, and Yunzhong He. Reward hacking in rubric-based reinforcement learning. *arXiv preprint arXiv:2605.12474*, 2026.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean Hendryx. Adaptive guidance accelerates reinforcement learning of reasoning models. *arXiv preprint arXiv:2506.13923*, 2025.

Jingjie Ning, Shanshan Zhong, Xiaochuan Li, Ji Zeng, and Chenyan Xiong. One run is not an idea: The implementation lottery in automated research. *arXiv preprint arXiv:2607.26587*, 2026a.

Xuying Ning, Dongqi Fu, Tianxin Wei, Hanqing Zeng, Yuanchen Bei, Bingxuan Li, Zihao Li, Qifan Wang, Xiang Shen, Yifan Wu, Jiayi Liu, Hong Li, Yinglong Xia, Xiangjun Fan, Hanghang Tong, and Jingrui He. EvoHarness-RL: Learning self-evolving runtime harness for long-horizon LLM agents. *arXiv preprint arXiv:2608.05446*, 2026b.

OpenAI, 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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *International Conference on Machine Learning (ICML)*, pp. 47717–47737, 2025.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. arXiv:2507.02833.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. POPE: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016. arXiv:1511.05952.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shobhita Sundaram, John Quan, Ariel Kwiatkowski, Kartik Ahuja, Yann Ollivier, and Julia Kempe. Teaching models to teach themselves: Reasoning at the edge of learnability. In *International Conference on Machine Learning (ICML)*, 2026.

Mirac Suzgun and Adam Tauman Kalai. Meta-prompting: Enhancing language models with task-agnostic scaffolding. *arXiv preprint arXiv:2401.12954*, 2024.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *The Thirteenth International Conference on Learning Representations*, 2025.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Beining Wang, Weihang Su, Hongtao Tian, Hao Kong, Tao Yang, Ting Yao, Qingyi Pan, Yueyue Wu, Qingyao Ai, Min Zhang, and Yiqun Liu. Co-evolving LLM evaluators and policies via DynamicRubric. *arXiv preprint arXiv:2607.20083*, 2026a.

Jianze Wang, Kunwang Zheng, Ying Liu, Yu Cao, Qilong Zhang, Jinlong Chen, Hua Yang, and Qianglong Chen. SERPO: Self-evolving rubric policy optimization for open-ended test-time reinforcement learning. *arXiv preprint arXiv:2607.26873*, 2026b.

Noah Wang, Z.y. Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhao Huang, Jie Fu, and Junran Peng. RoleLLM: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14743–14777, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.878. URL https://aclanthology.org/2024.findings-acl.878/.

Xintao Wang, Heng Wang, Yifei Zhang, Xinfeng Yuan, Rui Xu, Jen-Tse Huang, Siyu Yuan, Haoran Guo, Jiangjie Chen, Shuchang Zhou, Wei Wang, and Yanghua Xiao. CoSER: Coordinating LLM-based persona simulation of established roles. In *International Conference on Machine Learning (ICML)*, pp. 64822–64858, 2025.

Xuekang Wang, Zhuoyuan Hao, Shuo Hou, Hao Peng, Juanzi Li, and Xiaozhi Wang. Reproducing, analyzing, and detecting reward hacking in rubric-based reinforcement learning. *arXiv preprint arXiv:2606.04923*, 2026c.

Peilin Wu, Xinlu Zhang, Kun Wan, Wentian Zhao, Gang Wu, Xinya Du, and Zhiyu Chen. AMARIS: A memory-augmented rubric improvement system for rubric-based reinforcement learning. *arXiv preprint arXiv:2605.18592*, 2026a.

Yuhao Wu, Yushi Bai, Zhiqiang Hu, Roy Ka-Wei Lee, and Juanzi Li. LongWriter-Zero: Mastering ultra-long text generation via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026b. arXiv:2506.18841.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. WritingBench: A comprehensive benchmark for generative writing. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Can Xie, Yuyi Zhou, Wen Yang, Ziyi Zhang, Siyao Song, Yingzhuo Deng, Shuo Ren, and Jiajun Zhang. EDGE: Experience-distillation for guided exploration in agentic reinforcement learning. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tuo Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable LLM post-training. In *International Conference on Machine Learning (ICML)*, 2026.

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.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 50528–50652, 2024.

Qinyuan Ye, Yu Li, Yada Pruksachatkun, Jiaxin Zhang, and Chien-Sheng Wu. On the fragility of self-improving agents: Variance, task order, and underspecification. *arXiv preprint arXiv:2608.18066*, 2026.

Li S. Yifei, Allen Chang, Chaitanya Malaviya, and Mark Yatskar. ResearchQA: Evaluating scholarly question answering at scale across 75 fields with survey-mined questions and rubrics. *Transactions of the Association for Computational Linguistics*, 14:1365–1389, 2026. doi: 10.1162/tacl.a.732.

Linhao Yu, Tianmeng Yang, Siyu Ding, Renren Jin, Naibin Gu, Xiangzhao Hao, Shuaiyi Nie, Deyi Xiong, Weichong Yin, Yu Sun, and Hua Wu. KnowRL: Boosting LLM reasoning via reinforcement learning with minimal-sufficient knowledge guidance. *arXiv preprint arXiv:2604.12627*, 2026a.

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. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026b.

Yuanqiang Yu, Yanzhao Zheng, Zhentao Zhang, Tianze Xu, Chao Ma, Jihuai Zhu, Jiashun Liu, Xinle Deng, Baohua Dong, Hangcheng Zhu, and Ruohui Huang. PAC: Progress-augmented advantage curriculum for multi-task reinforcement learning of LLMs. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026c.

Zhou Yu, Bin Bi, Shiva Kumar Pentyala, Shubham Mehrotra, Sougata Chaudhuri, Shilpa Bhagavath, Zeyuan Chen, Ran Xu, Phil Mui, James Zhu, and Sitaram Asur. Co-evolving harnesses and models: On-policy correction helps weaker models catch up where imitation fails. *arXiv preprint arXiv:2609.09134*, 2026d.

Yongcheng Zeng, Zexu Sun, Bokai Ji, Erxue Min, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Haifeng Zhang, Xu Chen, and Jun Wang. CurES: From gradient analysis to efficient curriculum learning for reasoning LLMs. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.01037.

Qiyan Zhao, Xiaofeng Zhang, Bo Liu, Minda Chen, Wei Xiong, Jingyang Chen, Guanting Ye, Wenhao Yu, Xiaosong Yuan, Shijie Han, Da-Han Wang, Jianmin Ji, Fei Huang, and Xu-Yao Zhang. Coalition-aware skill reliability for self-evolving agents. *arXiv preprint arXiv:2608.22610*, 2026a.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *International Conference on Machine Learning (ICML)*, 2026b.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv:2604.08224*, 2026.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Kongcheng Zhang, Jiale Zhao, Jingwen Yang, Yihe Zhou, Jianwei Lv, Tongya Zheng, Hengtong Lu, Wei Chen, Yan Xie, and Mingli Song. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general LLM reasoning. *arXiv preprint arXiv:2508.16949*, 2025.

A LIMITATIONS

The harness adapts only task sampling, rollout guidance, and rubric criteria, while all other training settings remain fixed and its proposer is restricted to a closed vocabulary of operations. Failures requiring another setting or operation are therefore outside its reach. One open-weights judge family supplies the reward, evidence for each proposal, and the acceptance measurement, so acceptance is partly self-certifying (Liang et al., 2026). Independent out-of-distribution scorers bound but do not remove this circularity (Chen et al., 2026a; Tan et al., 2025; Dorner et al., 2025). Without donor eviction, stale updates remain retrievable after the policy has moved past them (Xie et al., 2026; Zhao et al., 2026a). Sensitivity of memory-bearing loops to sampling order and implementation variance (Ye et al., 2026; Ning et al., 2026a) also bounds claims that harness updates are invariant to either factor. The harness raises the per-step cost to $1.6\times$ that of GRPO, and the critic's measurements account for half of the added cost. We train two policies from one family, Qwen3.5-4B and Qwen3.5-9B, and use one 31B judge backbone for every harness role. This couples harness design to its operator's competence and confines the evidence to one policy family and scale range. All arms also train on one corpus, limiting the evidence to that training distribution. Main-text scores report the best checkpoint on each evaluation set and thus measure peak observed performance, while Appendix C reports the selection-free step-500 checkpoint and matched-compute results. The out-of-distribution suite uses one benchmark and one scoring protocol per domain, limiting each per-domain transfer claim to that pair. The failure classes use an oracle ranking from Claude Sonnet 5 under one quality guideline per domain, so the class proportions inherit that model's preferences. The attributor receives only the task, the current harness, the rollouts, their criterion scores and retrieved training evidence, limiting diagnosis to failures expressed through those inputs in the training corpus. It cannot identify a failure the training corpus never exposes, as in the MedQA case of Appendix C, a criterion that the judge consistently misreads, or a failure that emerges only in multi-turn agentic rollouts.

B METHOD DETAILS

Memory records. Each record m_t preserves the information needed to reconstruct one draw of a task, the harness decisions made at that draw, and their subsequent effects. It contains three fields. The `<rollout history>` field stores the task x_t, its rollout group Y_t, and the score matrix $S_t \in \{0,1\}^{K_t \times J}$ with entries $S_{t,mj} = s_{t,m}^{(j)}$. This field gives the evolver the policy outputs and criterion-level judgments that produced the current diagnosis. The `<rubric history>` field stores the harness $\mathcal{H}_t$ under which the rollouts ran, together with every action the evolver proposed at that step, whether accepted or rejected, and the measurement that decided it. This record allows a later proposal to distinguish an update that has not been attempted from one that was previously rejected or subsequently reversed. The `<training dynamics>` field stores the task's reward trajectory over the run and its harness update history, so that a later step can read how each earlier update changed the reward.

Retrieval. Feeding the full memory into every evolver decision is impractical because the history grows past the evolver's context and buries the records relevant to the current decision. Retrieval therefore selects evidence along two complementary axes. The time index fixes the task and varies the step, returning the task's records from earlier steps. These records tell the attributor whether a failure persists across steps and tell the proposer which updates have already been attempted on the same task. The semantics index instead fixes the criterion and varies the task. It returns the top-k criteria of other tasks whose embedding $\phi(\cdot)$ has cosine similarity at least η with the current criterion text, together with the positive and negative responses recorded for them. These cross-task records provide donor responses for the `<reference>` field and examples of how a similar criterion was previously rewritten. The two indexes consequently serve different decisions. The time index supplies task-specific training dynamics, while the semantics index supplies transferable evidence when the current task's own history is insufficient. Appendix D gives the index implementation and the floor η.

Evolver loop. The evolver separates diagnosis from proposal and acceptance so that revising an update does not also revise the failure attribution that motivated it. The attributor runs once per sampled task and uses the task, the current harness, the rollouts, their criterion scores, and the retrieved evidence, but never the oracle ranking or oracle quality, which are reserved for evaluation. Its diagnosis selects the interfaces to update. The proposer and critic then iterate for at most B rounds to determine which update, if any, should be accepted at each of them. Besides the diagnosis, the

retrieved records and the current harness, the proposer also reads the task x_t, the recorded group Y_t and its criterion scores S_t. At round r, the proposer receives $F^{(r-1)}$, the updates rejected in the previous round with the measurements that rejected them. It uses this measured feedback to revise the rejected updates or replace them with `noop`, while keeping the accepted ones unchanged. The loop does not return to the attributor because revising an action does not change the diagnosis that selected its interface. When the budget is exhausted, an interface with an unresolved update keeps its current setting, which prevents an untested or repeatedly rejected proposal from changing the harness.

Proposer rules. Three rules constrain the proposer so that its candidate updates remain measurable and preserve the training objective's scale. First, a training schedule lowers the number of guidance fields a task may hold by one at each quarter of training, from three to zero. An `add` is allowed only below the current cap, so that $\boldsymbol{g_t} = \emptyset$ for every task by the end of training and at evaluation. This rule permits privileged guidance when exploration is difficult but ensures that evaluation does not depend on that guidance. Second, anchor criteria, a designated subset of the seed criteria whose text is fixed at the memory cold start, allow `reweight` alone, which moves weight among anchors inside their fixed share. Every rubric operation renormalizes the adaptive weights so that the positive weights total Λ with the anchors at a fixed share, which keeps the anchors' share of the reward fixed as criteria change. The proposer writes small integer weights, and renormalization scales them so that the positive weights sum to Λ. Third, the operations proposed for one interface at the same step are applied together as one candidate update while the other two interfaces are held fixed. Each primary update is then measured at its own interface, so its measurement remains attributable to that interface. Secondary rubric operations join the rubric candidate of the same step.

JIT critic measurements. The critic evaluates each candidate update using the least expensive query that can measure its immediate effect, while holding the policy parameters fixed. For a rubric update, the critic regrades the recorded group Y_t under the candidate rubric $\widetilde{c}_{t+1}$, which costs judge calls but no policy calls. The resulting measurement isolates the scoring change induced by the candidate rubric because the responses remain unchanged. For a rollout update, it instead regenerates a fresh group $\widetilde{Y}_t$ under the candidate guidance at the same parameters θ_t, which costs one rollout group per round. This measurement tests whether the candidate guidance changes what the current policy can produce rather than merely changing how an existing group is scored. A sampling update needs no model query because a reweight changes only how often the task is sampled. The sampling update needs no band because a task reweight has no null variation to calibrate against. Here Δ_w, the reward-weighted sampling mass that the lower weight moves to other tasks, measures reallocated sampling probability rather than a change in the objective. Because a reweight never raises the weight, $\Delta_w > 0$ holds whenever it lowers a task with positive group reward, so the critic also requires the time index to show a `saturated` diagnosis of the task in its previous record, which rejects transient saturation. For a guidance update for a `blind` criterion or a rubric update for a `spurious` criterion, the memory cold start sets the band of the Magnitude rule separately for each interface (Appendix D). Its lower bound rejects updates indistinguishable from judge or sampling noise, and its upper bound rejects updates whose effect exceeds the typical group variation measured at the memory cold start. The Variation rule additionally rejects a `spurious` rubric update that leaves any criterion whose text it creates or rewrites constant across the group, because a constant criterion contributes nothing to the group-relative advantage. A guidance `drop` is proposed once its target leaves `blind` and measured on a fresh group without the field. It is accepted when $\Delta_g \geq -\varepsilon^g_{\text{low}}$, which shows that the policy no longer needs the scaffold.

C Additional analyses

`HarnessRL` retains its advantage at Qwen3.5-9B. The larger policy improves every method, raising GRPO to 63.6 in domain and 51.4 out of distribution and `HarnessRL` to 69.5 and 57.3. This persistent margin indicates that an adaptive harness retains headroom as the policy grows stronger. The family-level pattern also persists, with guidance-side methods gaining primarily in domain and rubric-side methods transferring further. At this scale, `HarnessRL` again reaches its best out-of-distribution checkpoint at step 250, against GRPO's 475.

`HarnessRL` reaches its best transfer in nearly half the steps GRPO needs. `HarnessRL` peaks at step 250, AMARIS++ at 300, EvoRubric at 350, RuscaRL at 425, and GRPO only at 475. This ordering results from the design because the harness corrects the reward signal as it degrades, whereas

GRPO continues training against a signal it never corrects. Out-of-distribution scores peak well before the end of training for every method that transfers, while in-domain scores keep rising to the last step for GRPO and every rubric-evolving method. We therefore report the best checkpoint per evaluation set in every table because endpoint reporting would understate every transfer claim, including those of the baselines. The gains also vary across domains. On the in-domain held-out set, `HarnessRL` is $+6.9$ pp above GRPO on medical and $+7.4$ pp on science, but only $+2.6$ pp on instruction following, where the anchor criteria are already programmatic.

Post-training matrix remeasurement identifies the source of the recovered signal. Repeating the diagnosis protocol on the trained checkpoints, the base run re-measures within half a point of the diagnosis. The `spurious` class splits into `off-axis`, in which the reward separates a pair that the oracle judge ties, and `inverted`, in which the reward order opposes the oracle ranking. The base run's 14.2 `off-axis` and 19.8 `inverted` become 11.4 and 15.8 under `HarnessRL`, so 4.0 of the 6.8 pp reduction in `spurious` comes from `inverted` pairs, which train the policy in the wrong direction. The `blind` and `saturated` proportions together shrink by 3.7 pp, and every recovered point lands in `agree`.

Which training-side metric ranks the arms? We rank-correlate each end-of-training class proportion with the arms' best-checkpoint scores, with failure-class proportions sign-flipped so that higher is better. **The `agree`, `saturated` and `spurious` proportions rank the arms on both evaluation sets, while the `blind` proportion alone predicts nothing.** Over the twelve arms with recorded class proportions, comprising the six arms of Table 1 other than SkillRL, the four additional arms of Table 3, and $\texttt{HarnessRL}_g$ and $\texttt{HarnessRL}_c$ without critic and memory, the Spearman correlations are 0.91 in domain and 0.92 out of distribution for `agree`, 0.59 and 0.74 for `saturated`, 0.78 and 0.82 for `spurious`, and 0.15 and -0.01 for `blind`. The arm with the lowest `blind` proportion, RuscaRL, transfers less than every rubric-evolving baseline, because helping the policy satisfy a frozen rubric converts `blind` into `saturated` rather than into aligned contrast. What ranks the arms on both evaluation sets is how little saturated and misdirected signal they carry, which is the pair of classes that the rubric-side arms lower and that the three-interface arms with memory lower without raising `blind`, `HarnessRL` the most.

Failure classes by arm. Every single-interface arm with recorded class proportions worsens a failure class assigned to an unavailable interface, and, of the ablation arms, only the three-interface arms with memory lower all three classes. All three guidance-side ablation arms, RuscaRL and $\texttt{HarnessRL}_g$ without and with critic and memory, increase saturation to 17.1, 16.5 and 16.1, because the improving policy outgrows a rubric that remains fixed. Conversely, all three rubric-side ablation arms, AMARIS and $\texttt{HarnessRL}_c$ without and with critic and memory, end `blind` at 13.3, 13.7 and 13.2, above the 12.0 of GRPO, because tightening the rubric without guidance leaves rollout groups unable to satisfy it. The `spurious` proportion falls by more than 2 pp only on the rubric side, reaching 28.2 at best, while the guidance-side arms end between 33.3 and 33.8, within 0.7 pp of GRPO's 34.0, because criteria that are never rewritten continue to measure the same properties incorrectly. The critic and memory reverse both gaps: without them the guidance-only and rubric-only harnesses reach 44.0 and 45.7, below RuscaRL's 44.3 and AMARIS's 46.5, and with them 46.3 and 47.2, above both. With all three interfaces, the same progression raises the harness from 44.1, within 0.2 pp of GRPO, to 45.9 with memory and then to 49.8 with the critic, surpassing AMARIS++ at 47.0. With all three interfaces, `HarnessRL` without the critic ends at 11.1, 15.2 and 31.6 (`blind`, `saturated`, `spurious`), and AMARIS++, whose only memory is its per-task revision log, ends at 11.4, 14.6 and 30.8, both below GRPO's 12.0, 15.6 and 34.0, while `HarnessRL` without critic and memory raises `saturated` to 16.1. `HarnessRL` is lowest at 10.5, 13.4 and 27.2. Memory therefore makes the evolver effective, and the critic adds a further $+3.9$ pp beyond memory.

Interface trajectories. Guidance and rubric updates concentrate early, while sampling changes continue to accumulate throughout training. Figure 4 traces the guidance and rubric trajectories. Mean guidance strength reaches 1.6 fields within the first 75 steps, and the attributor finds a `blind` criterion on 38% of sampled tasks in the first hundred steps. It then decays while the schedule still permits three fields, and 86% of tasks reach zero guidance before the schedule's last cut at step 375. Without the critic, guidance instead remains near the ceiling until each scheduled cut because nothing measures whether a task still needs its scaffold. Accepted rubric updates decline from $1{,}240$ in the first hundred steps to 140 in the last hundred, with `create` concentrated early and `rewrite` and `reweight` accounting for most later rubric updates. In the first hundred steps, 27% of tasks leave

full weight, and by step 500, 52% are below full weight, with 8% at the lowest level. The critic and memory limit reversed updates, where we count an accepted rubric update as a reversal when it undoes an earlier update on the same criterion within the task's previous two draws. The proportion of reversed updates reaches 22.3% without critic and memory and 18.2% for AMARIS++, rises from 6.8% to 14.1% of accepted rubric updates between the first and last hundred steps with memory alone, and falls from 3.1% to 1.6% under the full harness.

Sign of the rubric updates. Within the harness, the critic sets the sign of Δ_c on the gated rubric updates for `spurious` criteria, and without the critic every harness arm that updates the rubric loosens it. Δ_c is the immediate score drop that a rubric update induces on the recorded group, so a positive Δ_c tightens the rubric. We report its mean over the accepted rubric updates for `spurious` criteria in the last hundred steps, and over all accepted revisions in that window for AMARIS. Every rubric-updating harness arm without the critic has a negative Δ_c, with -2.0 for $\texttt{HarnessRL}_c$ without critic and memory, -3.2 for `HarnessRL` without the critic and -5.2 for `HarnessRL` without critic and memory. These updates make the recorded responses score higher, thereby writing reward inflation into the target. By contrast, $\texttt{HarnessRL}_c$ reaches $+6.8$ and `HarnessRL` reaches $+7.2$, both within the band imposed by the Magnitude rule, so these values give the margin of the gated updates rather than an independent test. AMARIS also tightens the rubric without a critic, since its log-driven revisions remove criteria the policy has already met, but by a smaller margin than either harness arm with the critic, $+4.8$, because nothing calibrates the tightening against the current group. Downstream progress depends on what the policy retains rather than on the measured gain: the retention rate of guidance rises from 0.32 without critic and memory to 0.62 on the full harness, the recovery rate after rubric tightening rises from 0.50 on AMARIS to 0.60, and they correlate with the in-domain trajectory at $r = +0.87$ and $+0.91$ while raw Δ_g and Δ_c do not, which is why the critic gates on the measurement but never maximizes it. Three annotators confirmed 86.0% of `spurious` labels and judged 82.5% of accepted rubric updates to address the diagnosed failure, and the proportion of pairs that the reward orders as the oracle judge does ends training at 48.9, compared with 38.4 under GRPO.

Harness dynamics. Guidance begins to decay after step 75 even though the scheduled ceiling still permits three fields, because guidance is withdrawn once its targeted criteria leave `blind` and the policy keeps its reward without it. A tightening rubric update can return those criteria to `blind` and thereby create a need for new guidance, so the interfaces are coupled through the state each one changes. Memory supplies cross-task evidence for these decisions. Of the `<reference>` fields accepted by the critic, 58% came from a cross-task donor rather than the task's own history. Proposals grounded in retrieved evidence were accepted at 71%, compared with 44% when retrieval returned nothing above the floor η.

Retention and recovery rates predict downstream performance, whereas raw measured gains do not. The retention rate is the fraction of Δ_g that survives after the guidance is removed, isolating the part written into θ rather than read from the context. The recovery rate is the fraction of the score drop Δ_c that the policy recovers by the task's next draw, and it is defined only where $\Delta_c > 0$. Across RuscaRL, the five guidance-updating harness arms (the four of Table 3 that update guidance and $\texttt{HarnessRL}_g$ without critic and memory), and the four arms that tighten the rubric (`HarnessRL`, $\texttt{HarnessRL}_c$, AMARIS and EvoRubric), at five checkpoints each, retention correlates with the in-domain trajectory at $r = +0.87$ and recovery at $r = +0.91$, whereas Δ_g itself correlates at only $+0.02$ and Δ_c at $+0.25$. A large measured gain that the policy does not retain indicates scaffolding rather than progress, which is why the critic gates on the measurement but never maximizes it.

Human annotators confirm both the attributor's labels and the rubric updates they trigger. Three annotators judged 200 attributor labels, 50 from each class, and 200 accepted rubric updates sampled uniformly from the Qwen3.5-4B run, blind to the training step and to the model's rationale. Mean pairwise Cohen's κ was 0.71 for labels and 0.68 for updates. Under majority vote, annotators confirmed 97.0%, 98.0%, 86.0% and 90.0% of the `blind`, `saturated`, `spurious` and `agree` labels and judged 82.5% of the updates to address the failure identified by the corresponding label, while most remaining updates modified criteria that annotators already judged correct.

Cost decomposition. The harness adds costs absent from GRPO, including attributor and proposer calls, critic regrading and fresh rollout groups, memory retrieval and writes, and a one-off memory cold start. Figure 5 decomposes each method's cost in cluster hours by stage and compares its best out-of-distribution score with the hours required to reach it. **The harness costs $1.6\times$ GRPO per step**

and still reaches its best out-of-distribution checkpoint in fewer cluster hours. Per 100 steps, GRPO spends 4.3 cluster hours on rollouts and judging, while the harness spends 6.7. The added 2.4 comprises 1.2 for the critic, 0.7 for attributor and proposer calls, 0.4 for memory, and 0.1 for longer rollouts under mounted guidance, which take 3.3 versus GRPO's 3.2 hours per 100 steps. A one-off 0.75-hour memory cold start is incurred separately. The harness peaks at step 250, whereas GRPO peaks at 475, so its best out-of-distribution checkpoint arrives after 17.5 hours, compared with GRPO's 20.4 and AMARIS++'s 18.3. The critic is therefore the largest single overhead, and the saving coincides with its measurements letting the harness withdraw guidance and cut accepted rubric updates from 1,240 to 510 per 100 steps by the middle of training.

`HarnessRL` remains strongest without best-checkpoint selection and under matched compute. Because best-checkpoint selection uses the evaluation sets, we also report the final step-500 checkpoint of every arm at Qwen3.5-4B, which requires no selection. In-domain/OOD scores are GRPO 57.8/43.8, RuscaRL 60.5/44.2, SkillRL 60.1/44.5, EvoRubric 60.7/45.7, AMARIS 60.9/46.0, AMARIS++ 61.5/46.4, and `HarnessRL` 64.5/49.4. `HarnessRL` therefore leads AMARIS++ by 3.0 points both in domain and OOD, compared with best-checkpoint margins of 3.0 and 2.8. At the 17.5 cluster hours that `HarnessRL` takes to reach its best OOD checkpoint, GRPO reaches 43.8 and AMARIS++ reaches 46.9, against 49.8 for `HarnessRL`.

Case study in full. Every update answers the shortcut the previous rubric rewarded, and a changing rubric decouples the group reward from policy quality. Figure 6 traces one chemistry task on the properties of cellulose filter paper through four of the steps at which the harness acted. A rollout first inflated the hydroxyl count to appear thorough, so at step 30 the harness turned the count into a checkable criterion, added two chemistry criteria beside it, and deleted the seed criterion on point-of-care use, since the task's two seed criteria are not anchors and may therefore be deleted or merged. At step 55, no rollout named the coupling reaction, so a `blind` label mounted a criterion field and a donor reference. By step 130, every rollout named the reaction and the criterion was `saturated`. The harness therefore merged wicking and the adsorption-versus-covalent criterion into a device-level synthesis criterion at weight $+3$, added pore-size and sensitivity criteria, deleted the name-reaction criterion that every rollout now satisfied, lowered the sampling weight to $\gamma_1 = 0.61$, and removed the two guidance fields. Synthesis remained `blind` while generic lists continued to score, so at step 260 the harness reweighted the device-level criterion and the hydroxyl count to $+1$, deleted the pore-size criterion that no rollout could satisfy and the sensitivity criterion that had turned `spurious`, restored the two chemistry criteria, and mounted a critique field, which the schedule's last cut removed at step 375. At step 440, once wicking and immobilization were satisfied, the harness created an intrinsic-versus-engineered criterion at $+2$ and, because the hydroxyl count had turned `saturated` again, lowered the sampling weight again to $\gamma_2 = 0.38$, after which every criterion except the hydroxyl count reads `agree`. Across these updates, the group reward falls from 100.0 at step 130 to 32.8 at step 260, while the oracle quality of the same groups rises from 58.1 to 61.0. The rubric changes between the two measurements, which is why the critic compares rubrics only on one recorded group at fixed θ and never treats group rewards from different steps as points on one scale.

Training-side failure modes differ from out-of-distribution failure modes. This distinction bounds what the harness can correct. On a MedQA item asking which outcome measure suits a study that uses a questionnaire and a single serological screen, the checkpoint at step 100 answers incidence, having seized on the word *acquisition* in the stem and supplied a longitudinal follow-up the study does not describe. From step 200 onward it answers prevalence and reasons from the study design. The knowledge was never missing. What was missing is the habit of letting the design constrain the keyword, which is expressible as a criterion but not observable on this corpus, whose medical prompts are open-ended generation rather than four-option questions. The failures visible to the harness and those scored by benchmarks overlap but differ, which explains why gains vary across benchmarks.

D EXPERIMENTAL DETAILS

Corpus and baselines. Medical tasks come from HealthBench (Arora et al., 2025) and the medical slices of RaR (Gunjal et al., 2026) and RubricHub (Li et al., 2026b). Science tasks come from ResearchQA (Yifei et al., 2026) and the matching slices of RaR and RubricHub, writing tasks come from LongWriter-Zero (Wu et al., 2026b), and role-play and instruction-following tasks come from

the chat and instruction slices of RubricHub (Table 2). The in-domain held-out set takes 250 tasks per domain from these same pools and is disjoint from training. GRPO (Shao et al., 2024) keeps all three interfaces fixed. RuscaRL (Zhou et al., 2025) and SkillRL (Xia et al., 2026) adapt rollout guidance while keeping the rubric fixed, whereas EvoRubric (Guan et al., 2026) and AMARIS (Wu et al., 2026a) revise the rubric without guidance. AMARIS++ extends AMARIS to all three interfaces by allowing its rubric analysis to reweight tasks and write guidance fields. It retains AMARIS's per-task rubric revision log as its only memory and accepts every proposal without measurement. It is therefore the closest baseline to our formulation.

Evaluation protocols. We use three evaluation protocols because a loop that updates its own reward can raise the score that it also defines. The in-domain held-out sets use the same rubric-scoring pipeline as training, so they measure performance under the training reward. Each held-out task is scored under its fixed seed rubric by the frozen judge, so no harness update reaches the evaluation rubric. The five out-of-distribution benchmarks use one benchmark per domain and change both the task distribution and the scoring protocol. Each is scored by its official protocol, which is exact choice for MedQA (Jin et al., 2021) and GPQA-Diamond (Rein et al., 2024), programmatic constraint checks for IFBench (Pyatkin et al., 2025), Rouge-L against references for RoleBench (Wang et al., 2024), and its own rubric for WritingBench (Wu et al., 2025). WritingBench alone calls the training judge, so the other four test whether gains persist beyond the reward that the harness updates. Because these protocols measure different quantities, absolute values are comparable only within a column. As the third protocol, we re-run the comparison of rubric reward with the oracle ranking of Section 3 on the trained checkpoints. This protocol scores the harness rather than the policy, so it reveals whether the reward has improved itself without improving agreement. Figure 1 reports, for GRPO, the mean oracle quality score of greedy decodes on the in-domain held-out prompts at every 25-step checkpoint, in a pass separate from the rubric-scored in-domain evaluation every 100 steps, and Figure 6 reports the mean oracle quality score over the recorded group at each step when the task was sampled.

Oracle judge. The oracle judge provides a semi-automatic evaluation reference through the oracle ranking for the failure classes and through oracle quality for the quality curves, and it never enters the harness as an input. Claude Sonnet 5 serves as the oracle judge. The oracle judge receives no rubric, reward or criterion score, but it is not criteria-free. Instead, it evaluates every response under one global quality guideline for its domain, whose principles apply to every task in that domain rather than to any single task. Medical responses are judged on clinical and factual correctness, safety through recognition of red flags and appropriate referral, advice the asker can act on, calibrated uncertainty, and clarity for the stated audience. Science responses are judged on correct facts, mechanisms and quantities, reasoning that connects evidence to claims, coverage at the depth requested by the question, and precise terminology. Writing is judged on fit to the task's purpose, genre and audience, coherent structure, quality of language and tone, and specific rather than generic content. Role-play is judged on consistency with the persona, engagement with the user's turn, narrative quality, and staying in character with appropriate content. Instruction following is judged on satisfying every explicit constraint, completing the core task, and adding nothing unrequested. In every domain, length, formatting and tone count only when the task requests them. For a group of $J = 8$ rollouts, the oracle judge compares all 28 pairs in both orders. Each call returns a preference or a tie. A pair's preference p_{ij} averages the two outcomes, with a win scored as 1 and a tie as 0.5, so $p_{ji} = 1 - p_{ij}$. The win counts $W_i = \sum_{j \neq i} p_{ij}$ determine the oracle ranking within the group, which places y_i above y_j when $W_i > W_j$ and ties them when $W_i = W_j$, and the pair-level classes in Section 3 compare each pair's reward order with this ranking. For the quality curves, the same pairwise prompt compares each response, in both orders, against eight fixed anchor responses per task that are sampled once from gemma-4-31B-it. The oracle quality score is 100 times the response's normalized win rate against these anchors, making scores comparable across steps and checkpoints. Two annotators independently ranked 240 sampled groups, and their rankings agreed at Kendall's $\tau = 0.84$. The oracle ranking agrees with the two annotators at 0.80 and 0.79, respectively, and with their curated consensus at 0.82, close to the agreement between the annotators.

Implementation. We train Qwen3.5-4B (Qwen Team, 2026) with GRPO (Shao et al., 2024) at group size $J = 8$ for 500 steps, using sampling temperature 1.0 and top-p 0.9, with 512 tasks per step, AdamW at learning rate 1×10^{-6}, KL coefficient $\beta = 0.001$, clip range $\epsilon = 0.2$ and $\epsilon_A = 10^{-6}$. We repeat every arm of Table 1 at Qwen3.5-9B on the full corpus under the same recipe. At evaluation, we decode greedily and provide no guidance, so nothing that the harness injects during training

is available when we score the model. One open-weights model, gemma-4-31B-it (Gemma Team, 2026), serves as the judge and as the backbone of the attributor, proposer and critic through a shared service pool. The judge grades each criterion at temperature 0.6. The judge role calls for exactly the discrimination that reward-model benchmarks measure (Lambert et al., 2025b; Malik et al., 2026),[1] and freezing and versioning the model prevents it from drifting during the run. We store the time index keyed by task and the semantics index in ChromaDB. The semantics index embeds criterion text with bge-base-en-v1.5 as ϕ and returns the top-$k = 4$ criteria whose cosine similarity clears the floor $\eta = 0.82$. We calibrate this floor so that paraphrases of one criterion clear it at 0.848, while similar wording that measures a different dimension remains below it at 0.792. Anchors change only by `reweight`. When a critic call fails, the update at that interface is rejected and the interface keeps its current setting, as when the round budget $B = 3$ is exhausted. The attributor and proposer prompts implement the labels and typed actions of Section 2.3, and the critic reuses the judge prompt. We report costs in cluster hours, the wall-clock hours of one 16-node cluster with eight NVIDIA H100 80GB GPUs per node.

Table 4: Seed statistics for the in-domain and out-of-distribution performance of `HarnessRL`, its ablations, and the baselines.

	Qwen3.5-4B		Qwen3.5-9B	
Method	In-domain	OOD	In-domain	OOD
GRPO	57.8 ± 0.4	43.9 ± 0.5	63.6 ± 0.4	51.4 ± 0.4
RuscaRL	60.8 ± 0.6	44.3 ± 0.7	65.8 ± 0.6	51.7 ± 0.6
SkillRL	60.4 ± 0.6	44.6 ± 0.7	65.4 ± 0.5	52.0 ± 0.6
EvoRubric	60.7 ± 0.7	46.2 ± 0.8	65.9 ± 0.6	53.5 ± 0.7
AMARIS	60.9 ± 0.6	46.5 ± 0.7	66.0 ± 0.5	53.8 ± 0.7
AMARIS++	61.5 ± 0.6	47.0 ± 0.7	66.6 ± 0.6	54.2 ± 0.6
`HarnessRL`	64.5 ± 0.3	49.8 ± 0.4	69.5 ± 0.2	57.3 ± 0.3
w/o critic	–	45.9 ± 0.8	–	–
w/o critic & mem.	–	44.1 ± 0.9	–	–
$\texttt{HarnessRL}_g$	–	46.3 ± 0.5	–	–
$\texttt{HarnessRL}_c$	–	47.2 ± 0.5	–	–

Seeds and statistical tests. Every arm is trained with three seeds. Every table reports the seed mean under the checkpoint protocol of Section 3, and the curves in Figures 3 and 4 show seed means. Table 4 reports the standard deviations, which remain below one point for every arm. `HarnessRL` has the smallest at both scales (0.3 in domain and 0.4 OOD at Qwen3.5-4B, 0.2 and 0.3 at Qwen3.5-9B), consistent with the critic rejecting updates whose effect falls outside the calibrated band. `HarnessRL` exceeds AMARIS++ on every seed at both scales. A paired bootstrap over evaluation items and seeds with 10,000 resamples gives 95% intervals for the margin over AMARIS++ of $[2.4, 3.6]$ in domain and $[2.0, 3.5]$ OOD at Qwen3.5-4B, and $[2.2, 3.6]$ and $[2.3, 3.9]$ at Qwen3.5-9B. A paired permutation test gives $p < 0.001$ for each margin and for every margin over GRPO. Removing both the critic and memory yields 44.1 ± 0.9 OOD, within one standard deviation of GRPO's 43.9 ± 0.5, so these two modules carry the gain reported in RQ1.

Memory cold start. Before training, we calibrate the harness hyperparameters once on the entire training set $\mathcal{D}$, with the five domains pooled. Evaluation scores the policy alone, without the harness, on held-out and OOD sets disjoint from $\mathcal{D}$. We calibrate with the Qwen3.5-4B seed policy, and the Qwen3.5-9B runs reuse these settings. For each $x_i \in \mathcal{D}$, we draw J rollouts under the seed harness $\mathcal{H}_0$ and grade them into S_i^0. We first use these grades to set the sampling levels. For each task, we compute its *discriminative mass* $\iota_i = \frac{1}{K_0}|\{m : 0 < \sum_j S_{i,mj}^0 < J\}|$, which is the proportion of criteria that separate the group. EvoRubric (Guan et al., 2026) applies this test to prune criteria after training, whereas we apply it before training to fix the sampling scale. We initialize the task weights at $w_0^i = 1$, which gives uniform initial sampling, and set $\gamma_\ell = \max\big(\gamma_{\min}, \mathrm{median}\{\iota_i : \iota_i \in T_\ell\}\big)$ for the descending tertiles T_1, T_2, T_3. Because adjacent tertiles can have equal medians, the resulting levels satisfy only the weak ordering $\gamma_1 \geq \gamma_2 \geq \gamma_3$. We therefore collapse coinciding levels on a degenerate bank. The positive floor $\gamma_{\min} > 0$ keeps every sampling probability positive and u_t finite. Each accepted sampling update for a `saturated` task moves the task one level down, from

[1] `https://theaggregate.ai/model?slug=gemma-4-31b-it`

$w^i = \gamma_{\ell-1}$ to γ_ℓ with $\gamma_0 = 1$, so a task reaches γ_3 only after three sampling updates and a reweight never raises a weight. We next calibrate the acceptance bands separately for the rollout and rubric interfaces because fresh generation and judge regrading have different null variation. We define $\nu^g_{0.95}$ as the 95th percentile of $|\bar{r}(Y, c_0) - \bar{r}(Y', c_0)|$ over independent groups Y, Y' of the same task. We define $\nu^c_{0.95}$ as the same percentile of $|\bar{r}(Y, c_0) - \bar{r}'(Y, c_0)|$ over repeated gradings of one group. We measure the typical within-group reward range as $\varsigma = \mathrm{median}_i(\max_j r_{ij} - \min_j r_{ij})$, where $r_{ij} = r(x_i, y_i^{(j)}, c_0)$. We then set $\varepsilon^g_{\mathrm{low}} = \nu^g_{0.95}$, $\varepsilon^c_{\mathrm{low}} = \nu^c_{0.95}$ and $\varepsilon^g_{\mathrm{high}} = \varepsilon^c_{\mathrm{high}} = \varsigma$. On $\mathcal{D}$, this calibration gives $\nu^g_{0.95} = 10.2$, $\nu^c_{0.95} = 3.1$ and $\varsigma = 28.0$, together with sampling levels 0.61, 0.38 and 0.17 above $\gamma_{\min} = 0.05$. The interface-specific lower bounds ensure that an update with no true effect is accepted in any one measurement with probability at most 0.05, since $|\Delta|$ must exceed the corresponding null quantile. The shared upper bound caps an accepted update at the reward range already produced by the seed policy. Finally, the division by the total positive weight in Section 2.1 fixes the reward scale, and to keep the anchors' influence fixed we hold the rubric's total positive weight at a constant Λ, keep the anchors at a fixed share, and renormalize the adaptive weights to the remainder after every rubric update. We set $\Lambda = 100$, which makes $\Lambda\bar{r}$ the group's mean score in reward points, and we report every Δ and band in those points. We do not re-estimate the bands during training because doing so would move their boundaries with the rubric and confound a change in an update's effect with a change in the measuring scale.

E PROMPT TEMPLATES

This section provides the prompt templates for the harness roles and the oracle judge. gemma-4-31B-it serves as the judge and as the backbone of all three evolver roles, and the JIT critic has no prompt of its own because it calls the judge prompt to regrade a group and the policy to regenerate one. Claude Sonnet 5 serves as the oracle judge outside training, and no harness role sees its output. Fields enclosed in braces are populated at run time from the task, the current harness, the recorded group and its criterion scores, the attributor's labels, the rejected updates, and the retrieved memory records, while the block structure comprising the system role, inputs, instructions, and output schema is adapted from AMARIS (Wu et al., 2026a) with our label and operation vocabulary.

Harness attributor

System

You are the attributor of a training-time harness for rubric-based reinforcement learning. For one sampled task, you read the rollout group, the judge's criterion scores and the task's retrieved training evidence, and you diagnose why each criterion's signal succeeds or fails. One step's scores are insufficient because the same count can reflect different training states. A zero count may indicate that the policy cannot yet reach the criterion or that the current group is noisy. A full count may indicate that the policy has mastered the task. A criterion that separates the group may track a surface form rather than the intended property. Resolve these cases by comparing each criterion's current scores with the task's earlier records and with similar criteria from other tasks. Distinguish compliance with a criterion's wording from evidence that a rollout exhibits the property the criterion is intended to reward. First collect the evidence, then assess it along three dimensions and the criterion's wording, and only then assign a label. A fixed rule routes each diagnosis to one interface, so you do not write updates. You receive no quality signal beyond the judge's criterion scores. Base every statement on the input packet alone, cite rollouts by their ids, and output only valid JSON.

Inputs

`{task}` The task text without the guidance fields.
`{rubric}` Criteria C1 to CK, each with its text, a signed weight (a negative weight marks a penalty) and an anchor flag.
`{harness}` The task's current sampling weight and the guidance fields mounted for this group (`<criterion>`, `<reference>`, `<critique>`), with their content.
`{rollouts}` The group y1 to y8.
`{scores}` The judge's binary score for every criterion on every rollout, with each criterion's count k out of 8 rollouts that satisfy it (for a penalty criterion, those without the penalized behavior) and their ids.
`{time evidence}` This task's earlier records from the time index: each criterion's count and label at earlier steps, the updates proposed at those steps with their acceptance or rejection and measurement, and the task's reward trajectory.
`{semantic evidence}` For each criterion, up to 4 criteria from other tasks in the semantics index with cosine similarity of at least 0.82, each with its similarity, source task, one positive and one negative response, and earlier rewrites.

Part 1. Evidence

1. *Current scores.* Copy k and the ids of the satisfying and failing rollouts from `{scores}`. Never estimate them. For a penalty criterion, satisfied means that the penalized behavior is absent, so k counts the rollouts that the judge scores 0.
2. *Rollout quotes.* When $0 < k < 8$, quote the span in a satisfying rollout that earns the criterion and state what each cited failing rollout lacks.
3. *History.* From `{time evidence}`, record the criterion's earlier counts and labels, the updates proposed for it and their outcomes, and the task's reward trajectory.
4. *Neighbors.* From `{semantic evidence}`, record the neighbor criteria and their similarities. Note any surface form that separated their groups and how the criteria were rewritten.

Part 2. Assessment dimensions

1. *Reach, at $k = 0$.* Ask whether the policy can produce the property at all. Treat the current zero as a reach failure if the criterion has also remained unmet across the task's earlier records, since this persistence indicates that the current policy cannot yet reach it. If earlier counts were nonzero, treat noise as more likely than a reach failure. If no earlier record exists, the current zero is the only evidence and may support a `blind` label only with low confidence.
2. *Training stage, at $k = 8$.* Ask whether the policy has mastered the task. A criterion that every rollout meets across recent steps is mastered and no longer separates the group. As more criteria reach this state, the task's discriminative mass, the proportion of its criteria that still separate the group, declines. Report the current mass and its recent trend. Because sampling acts on the whole task, attribute saturation to the task rather than to one criterion. When every rollout meets a criterion through generic content, such as a disclaimer or a polite closing, the criterion demands less than the task's goal. State in the critique how it could demand more of this task, such as tighter reasoning for a science task, a more precise claim for a medical task, or closer adherence to the requested style for a writing task, so the proposer can rewrite it as a secondary rubric operation.
3. *Reward hacking, at $0 < k < 8$.* Ask whether meeting the criterion reflects the intended property or only a surface form. Compare the satisfying rollouts with the failing rollouts and with the neighbors' positive responses, which exhibit the property without that form. If a surface form separates the group, name exactly one of four families. (i) Superficial compliance restates the criterion's words, stuffs its keywords, claims compliance, or copies a required format without its substance. (ii) Length gaming

pads the response with verbosity or extra items presented as detail. (iii) Partial coverage meets one part of a compound criterion or mentions the topic without making the required claim. (iv) Stance gaming flatters the user, agrees with the user's premise, or refuses or hedges a benign request to meet a tone or safety criterion. Do not treat a length, list, format, or tone that the task itself requests as gaming. Check whether the surface form grew with the task's reward over recent steps, which indicates that the policy is learning it. Check whether a neighbor separated on the same form and how its criterion was rewritten, and cite the neighbor without copying its label.

4. *Rubric wording.* For a criterion separated by a surface form or met by every rollout through generic content, identify the exact phrase in its text that permits the surface form or sets a requirement the group already meets. Such phrases include a compound requirement, an open list of examples, wording that a mention can satisfy, or a property exhibited by any response. Use this phrase as the reason in the critique.

Part 3. Decision

1. *Label.* Label a criterion `blind` when $k = 0$ and either its earlier counts have also remained at 0 or no earlier record exists, in which case confidence must be low. Label it `saturated` when every rollout has met it across recent steps. Label it `spurious` when one named surface form, rather than the intended property, separates the group and is absent from the neighbors' positive responses. Label it `agree` otherwise, including when the history contradicts the current count or the surface-form evidence is weak. An `agree` criterion needs no update.
2. *Task diagnosis.* Diagnose the task as `saturated` when its `saturated` criteria have lowered its discriminative mass over recent steps. The diagnosis attaches to the task because the sampling weight acts on the whole task.
3. *Confidence.* Set each confidence from persistence: high when the criterion shows the same count pattern across the task's earlier records, and for `spurious` when the form grew with the task's reward, low without an earlier record. The JIT critic does not use this confidence for acceptance.
4. *Critique.* Describe this group's shortfall using rollout ids and quotes, so the critique can later populate a `<critique>` field. State the reason in one domain-agnostic sentence about the criterion's structure, naming the phrase found by the rubric wording check when one exists, for example, "The criterion checks a mention of the topic rather than the stated claim."
5. *Route.* A fixed rule sets the interface. A `blind` criterion routes to rollout guidance, and a `spurious` criterion routes to the rubric. A `saturated` or `agree` criterion routes to none, while a `saturated` task routes to task sampling.

Output format

```
{"criteria": [
  {"id": "C3",
   "evidence": {
     "satisfied": "k/8 with ids, copied from {scores}",
     "quotes": [{"rollout": "y2", "quote": "span that earns it"}],
     "lacking": [{"rollout": "y5", "lacks": "what it lacks"}],
     "history": "earlier counts, labels and updates, or none",
     "neighbors": ["neighbor id, separating form, rewrite"]},
   "checks": {
     "reach": "... or null", "training stage": "... or null",
     "reward hacking":
       "superficial compliance | length gaming | partial coverage |
        stance gaming | null",
     "growth": "the surface form's trend against reward, or null",
     "rubric wording": "the criterion phrase at fault, or null"},
   "decision": {
     "label": "blind | saturated | spurious | agree",
     "confidence": "low | medium | high",
     "critique": "what the group did, with rollout ids and quotes,
       and how a generically met criterion could demand more",
     "reason": "one domain-agnostic sentence on the structure",
     "route": "rollout | rubric | none"}}
 ],
 "task": {"mass": "proportion of separating criteria and its trend",
          "label": "saturated | none",
          "confidence": "low | medium | high",
          "route": "sampling | none"}}
```

Harness proposer

System

You are the proposer of a training-time harness for rubric-based reinforcement learning. Given the attributor's labels for one sampled task, you propose one update for each of three interfaces, task sampling, rollout guidance and the rubric, as a set of typed operations or `noop`. The JIT critic measures each update against the current policy and accepts or rejects it, so propose only updates whose effect it can measure. Base every update on the input packet alone and output only valid JSON.

Inputs

`{labels}` For each criterion, the attributor's label, confidence, critique, reason, route, checks, and quoted evidence, and for the task, its discriminative mass and trend together with its `saturated` diagnosis.
`{task}` The task text, without the guidance fields.
`{rubric}` Criteria C1 to CK with text, signed weight and anchor flag.
`{harness}` The task's sampling weight (1, 0.61, 0.38 or 0.17) and its mounted guidance fields with their content.
`{guidance cap}` The number of guidance fields a task may hold at this stage of training, 3, 2, 1 or 0 by training quarter.
`{rollouts, scores}` The group y1 to y8 and its criterion scores.
`{time evidence}` This task's earlier records, including the updates proposed there, their measurements, and whether they were accepted, rejected or later reversed.
`{semantic evidence}` Up to 4 neighbor criteria of other tasks per criterion with their positive and negative responses and earlier rewrites.
`{round}` The round r out of 3.
`{rejected}` The updates rejected in round $r-1$, each with its measurement and the condition it failed, and the updates already accepted. Empty in round 1.

Instructions

1. *Route by label.* Use a sampling update for a `saturated` task, a guidance update for a `blind` criterion, and a rubric update for a `spurious` criterion. An `agree` criterion needs no update, and an interface with no routed label receives `noop`. The same step may add rubric operations that delete, reweight or rewrite a `saturated` or `blind` criterion or create one to supplement it, and a guidance `drop` of a field whose target criterion has left `blind`.
2. *Sampling.* `reweight` lowers the task weight by one level along 1, 0.61, 0.38, 0.17 and never raises it. At 0.17 the only option is `noop`. The critic accepts a reweight only when the task's previous record also diagnoses the task as `saturated`.
3. *Guidance.* `add(f)` mounts one field `f` only while the task holds fewer fields than `{guidance cap}`, and `drop(f)` removes a field once its target criterion has left `blind`. Fill `<criterion>` with a constraint paraphrased from the `blind` criterion or with earlier behavior that satisfied it, `<reference>` with one positive and one negative response from the task's history or from a neighbor in `{semantic evidence}`, and `<critique>` with the previous group's shortfall taken from the attributor's critique.
4. *Rubric operations.* Use `create`, `rewrite`, `delete`, `reweight`, `merge` and `split` only. Anchor criteria accept `reweight` alone. Write small signed integer weights, a negative weight for a penalty, and let renormalization scale the positive weights to a total of 100 with the anchors at a fixed share.
5. *Rubric update for a* `spurious` *criterion.* Remove the surface form the attributor identified, whether superficial compliance, length gaming, partial coverage, or stance gaming. Split a criterion that joins several requirements into separate ones, state a required count instead of an open list of examples, require the exact claim together with its reason, or create a penalty criterion that describes the surface form.
6. *Criterion text.* Describe a property that the response text alone can show. Write a penalty as the failure itself ("appends an assessment of its own quality"), not as "fails to do X". Do not state facts the task does not give, never refer to a specific rollout, and do not reward length, format, tone or agreement with the user unless the task asks for them.
7. *Rubric hygiene.* Prefer fewer and clearer criteria. Do not duplicate an existing criterion or an anchor, and merge two criteria only when a response passes or fails both for the same reason.
8. *History.* Before proposing an update, look in `{time evidence}` for a similar update that was rejected or reversed on this task, and propose it again only with a stated difference. When a surface form returns after an earlier rewording, prefer a penalty criterion or a larger penalty weight over another rewording. List what you deliberately did not reintroduce.
9. *Measurement.* A rubric update for a `spurious` criterion must lower the recorded group's mean reward by 3.1 to 28.0 points and keep every criterion whose text it creates or rewrites varying within the group. A guidance `add` for a `blind` criterion must raise the mean reward of a fresh group by 10.2 to 28.0 points. A `drop` is accepted when a fresh group without the field loses at most 10.2 points.

10. *Revision.* From round 2 on, keep accepted updates unchanged, revise each rejected update using its measurement or replace it with `noop`, and do not change the labels.

Output format

```
{"sampling": {"rationale": "...",
   "operations": "noop" or [{"op": "reweight", "level": 0.61}]},
 "rollout": {"rationale": "...",
   "operations": "noop" or
     [{"op": "add | drop",
       "field": "criterion | reference | critique",
       "target": "C4", "content": "text of the field"}]},
 "rubric": {"rationale": "...",
   "operations": "noop" or
     [{"op": "create | rewrite | delete | reweight | merge | split",
       "target": ["C3"],
       "text": "criterion text without weight or id",
       "weight": 2, "reason": "..."}]},
 "not reintroduced": ["update shown as rejected or reversed"],
 "record": {"changed": "...", "lesson": "..."}}
```

Criterion judge

System

You grade one response against one criterion of a rubric. Judge this criterion alone and do not consider how good the response is overall. Output only valid JSON.

Inputs

`{task}` The task text, without any guidance fields.
`{response}` One response to the task.
`{criterion}` The text of one criterion, without its weight.

Instructions

1. Decide whether the response meets the criterion as written.
2. When the criterion states several requirements, it is met only if every one is met. Examples introduced by "such as" or "for example" need not all appear.
3. A partial match counts as not met, and a claim counts only when the response states it, not when it can be inferred.
4. When the criterion describes an undesirable behavior, it is met if the behavior is present. Report whether the criterion is met, not whether the response is good.
5. Give a brief reason before the answer.

Output format

```
{"reason": "one or two sentences", "met": 1 | 0}
```

Oracle judge, pairwise comparison

System

You compare two responses to one task under the quality guideline for the task's domain. You receive no rubric, reward or criterion score, and you judge the responses only by that guideline. Output only valid JSON.

Inputs

`{domain guideline}` The global quality guideline for the task's domain, shared by every task in that domain and listed below.
`{task}` The task text.
`{response A}` The response listed first.
`{response B}` The response listed second.

Domain guidelines

`{medical}` Clinical and factual correctness, safety that recognizes red flags and refers or escalates when needed, advice that the asker can act on in their situation, calibrated uncertainty without overclaiming, and clear communication for the stated audience.
`{science}` Correct facts, mechanisms and quantities, reasoning that connects evidence to claims, an answer that addresses what is asked at the requested depth, and precise terminology with calibrated uncertainty.
`{writing}` Fit to the task's purpose, genre and audience, coherent structure and development, quality of language, style and tone, and specific, original content rather than generic filler.
`{role-play}` Consistency with the character's persona, knowledge and voice, engagement with the user's turn and the scene, narrative quality and creativity, and staying in character while keeping the content appropriate.
`{instruction following}` Every explicit constraint met, including format, length, content and language, the core task completed correctly, no unrequested content or broken constraint, and as much help as the constraints allow.

Instructions

1. Judge which response better fulfills the task under every principle of `{domain guideline}`. Apply no task-specific rubric or checklist, and impose no requirement that the task and guideline do not state.
2. Do not prefer either response for its length, formatting or tone unless the task requests them. Do not let presentation order affect your judgment, because the caller judges every pair in both orders.
3. Answer `tie` only when neither response is better under the guideline.
4. State the differences that determine the preference before giving it.

Output format

```
{"reason": "the differences that decide the preference",
 "preferred": "A | B | tie"}
```