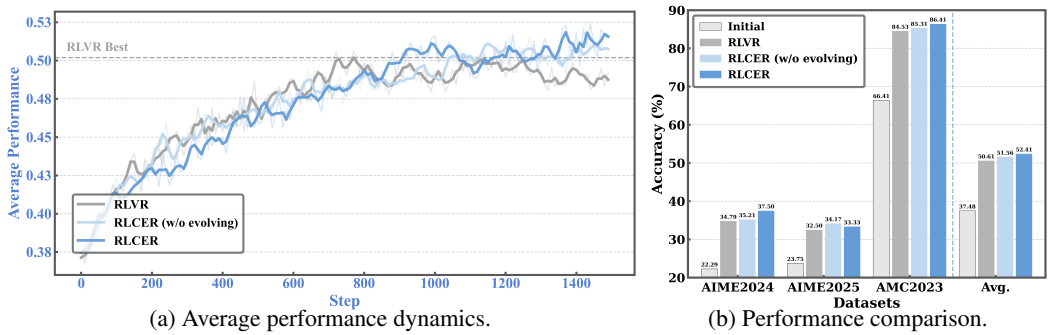
REINFORCING CHAIN-OF-THOUGHT REASONING WITH SELF-EVOLVING RUBRICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite chain-of-thought (CoT) playing crucial roles in LLM reasoning, directly rewarding it is difficult: training a reward model demands heavy human labeling efforts, and static RMs struggle with evolving CoT distributions and reward hacking. These challenges motivate us to seek an autonomous CoT rewarding approach that requires no human annotation efforts and can evolve gradually. Inspired by recent self-evolving training methods, we propose **RLCER** (**R**einforcement **L**earning with **C**oT Supervision via Self-**E**volving **R**ubrics), which enhances the outcome-centric RLVR by rewarding CoTs with self-proposed and self-evolving rubrics. We show that self-proposed and self-evolving rubrics provide reliable CoT supervision signals even without outcome rewards, enabling RLCER to outperform outcome-centric RLVR. Moreover, when used as in-prompt hints, these self-proposed rubrics further improve inference-time performance.



(a) Average performance dynamics. (b) Performance comparison.

Figure 1: Performance across three math datasets on the 8B model. Training with RLCER leads to a higher performance ceiling, and the self-evolving rubrics further enhance reasoning performance.

1 INTRODUCTION

Chain-of-thought (CoT) (Wei et al., 2022) reasoning is essential for large language models (LLMs) to solve complex tasks, where its quality strongly affects the final-answer correctness (Chang et al., 2025; Feng et al., 2025; Jiang et al., 2025; Zheng et al., 2025). However, CoT is rarely rewarded or optimized explicitly, even often regarded as a by-product of outcome-centric optimization objectives in large-scale reinforcement learning with verifiable rewards (RLVR) (DeepSeek-AI et al., 2025; Jaech et al., 2024; Shao et al., 2024). This yields an underconstrained learning signal: for the same final answer, many distinct reasoning trajectories receive identical rewards, allowing optimization to drift toward shortcut or brittle strategies (Zhou et al., 2025; Zhang et al., 2025a). Consequently, the model can converge to sub-optimal reasoning behaviors, limiting robustness and overall capability (Lightman et al., 2024).

However, while important, directly rewarding or supervising CoTs remains challenging in practice, mainly for two reasons: (i) training additional reward models (RMs) for CoT supervision typically requires labor-intensive and fine-grained annotations (Lightman et al., 2024; Cui et al., 2025), and (ii) the policy's CoT distribution shifts during training, yielding non-stationary and potentially biased supervision. ***Can the policy model self-propose rubrics as CoT supervision criteria and self-evolve***

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.

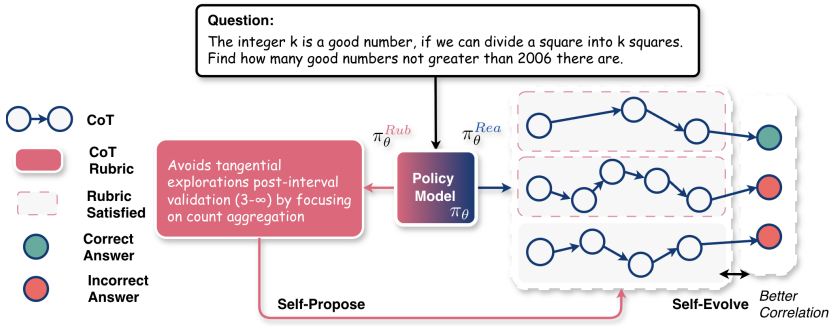


Figure 2: Key idea of reinforcement learning with CoT supervision via self-evolving rubrics (RLCER). The policy model π_θ acts as both the reasoner and the rubrics generator, self-generating and self-evolving the rubrics for CoT supervision, where the evolving direction is shaped towards the correlation with the rubrics satisfaction and the final answer correctness.

them during training with no human annotations? If so, this could enable a new paradigm of self-improving reasoning, shifting RL from optimizing what LLMs answer to autonomously improving how LLMs think, potentially yielding "free-lunch" reasoning gains.

We find possibilities of achieving this from the recent advances in the research line of self-evolving training methods (Gao et al., 2025; Silver et al., 2017), where a policy model π_θ can improve itself with little to no human intervention during training. On the one hand, π_θ itself can self-generate training data, such as verifiable question-answer pairs for RL or high-quality CoT trajectories for cold-starting reasoning (Zhao et al., 2025; Liu et al., 2025a; Lu et al., 2025; Zelikman et al., 2022). On the other hand, π_θ itself can self-assign reward signals by acting as a judge to evaluate its own solutions (Ye et al., 2025; Yuan et al., 2024; Fang et al., 2025; Chen et al., 2024). A key idea shared by these methods is to let a single policy model π_θ play multiple roles under different prompts, serving not only as a reasoner that produces solutions but also as a data generator or a reward provider (Zhao et al., 2025; Yuan et al., 2024), with all roles optimized jointly (*e.g.,* via techniques such as multi-agent reinforcement learning (Zhao et al., 2025)). Together, these capabilities make it plausible to self-propose supervision criteria (*e.g.,* rubrics) for assessing CoT quality and to self-evolve as training progresses. However, how to adaptively propose and evolve such criteria in the self-evolving training paradigm remains largely underexplored.

To fill this gap, we propose **RLCER** (**R**einforcement **L**earning with **C**oT Supervision via Self-**E**volving **R**ubrics), empowering RLVR with self-proposed and self-evolving rubrics to supervise CoT reasoning. The core idea is to let the policy model π_θ self-propose CoT supervision criteria as natural-language rubrics (*i.e.,* describing desirable reasoning properties like "avoid tangential explorations") (Gunjal et al., 2025; Zhang et al., 2025c), rewarding CoTs by their rubric satisfaction, and simultaneously improving the model's rubric-generation ability throughout training. An overview of the key process is illustrated in Figure 2. Specifically, a single policy model π_θ is instantiated with different prompts to play two roles: a reasoner π_θ^{Rea} that generates CoTs $\hat{\mathcal{C}}$ and final answers $\hat{\mathcal{A}}$, and a rubricator π_θ^{Rub} that generates a set of candidate rubrics $\hat{\mathcal{R}} \triangleq \{\hat{\tau}_k\}_{k=1}^K$, where K is the number of rubrics $\hat{\tau}_k$. The reasoner is rewarded by both the final answer correctness and the CoT quality, which is measured by satisfaction of all valid rubrics $\hat{\tau}_k^{valid}$. Here, we deem a rubric $\hat{\tau}_k$ valid if its satisfaction indicator $\mathbf{v}_k$ is strongly correlated with final-answer correctness $\mathbf{z}$ across multiple sampled rollouts for the same question $\mathcal{Q}$ (*i.e.,* $\texttt{corr}(\mathbf{v}_k, \mathbf{z}) > \alpha$), since a high correlation means that satisfying the rubric aligns with producing correct answers. Additionally, we encourage the rubricator to self-evolve by rewarding the fraction of valid rubrics among its proposals (*i.e.,* $|\{\hat{\tau}_k^{valid}\}|/K$).

Generally, RLCER enables RLVR to move beyond outcome-centric optimization by rewarding CoTs with supervision criteria that are self-proposed and self-evolved, without any human annotation. We further conduct comprehensive experiments to examine the effectiveness of the proposed RLCER. First, we conduct a primary study showcasing that self-proposed rubrics can be reliable and learnable even when training is driven solely by rubric-based rewards, without any outcome reward (*cf.* Section 4.2). Based on this insight, RLCER generally outperforms the naive RLVR (*cf.* Section 4.3). Additionally, introducing self-evolving training for the rubricator further improves the reasoning performance (*cf.* Section 4.4). Moreover, the generated rubrics can even serve as in-prompt hints to

enhance reasoning at inference time (*cf.* Section 4.5). Overall, our results suggest a new paradigm in which models autonomously generate CoT supervision criteria to strengthen RL training, pointing toward more self-improving and self-evolving reasoning systems.

2 PRELIMINARIES

In this section, we briefly review the background of our work, including incentivizing LLM reasoning with RL in Section 2.1 and multi-role RL under a single policy in Section 2.2.

2.1 INCENTIVIZING LLM REASONING WITH RL

In this work, we focus on incentivizing LLMs to reason with RL, where the policy model is optimized by rewarding the high-quality rollouts.

Rollout. Given a question $\mathcal{Q}$, the policy model π_θ attempts to solve it by generating a response that contains a CoT $\hat{\mathcal{C}}$ followed by an answer $\hat{\mathcal{A}}$, which can be formulated as:

$$\hat{\mathcal{C}}, \hat{\mathcal{A}} \sim \pi_\theta(\cdot \mid \mathcal{Q}). \tag{1}$$

Rewarding. After rollouts generation, rewards are assigned based on response quality, which can depend on: (i) the CoT $\hat{\mathcal{C}}$ and (ii) the answer $\hat{\mathcal{A}}$. In the widely adopted RLVR paradigm, rewards typically depend solely on the correctness of the predicted answer $\hat{\mathcal{A}}$:

$$r = \psi(\mathbb{I}(\mathcal{A}, \hat{\mathcal{A}})), \tag{2}$$

where $\mathbb{I}(\cdot, \cdot)$ is an answer correctness indication function which returns 1 if two answers $\mathcal{A}$ and $\hat{\mathcal{A}}$ are equivalent and 0 otherwise, and a reward assignment function ψ maps this correctness signal $\mathbb{I}(\mathcal{A}, \hat{\mathcal{A}})$ to a binary reward r. Through such RLVR training, the policy model π_θ gradually learns to answer questions, which is often associated with producing longer CoTs (DeepSeek-AI et al., 2025). However, under the RLVR paradigm, the CoT $\hat{\mathcal{C}}$ itself is rarely rewarded directly due to practical implementation challenges (DeepSeek-AI et al., 2025), while the lack of such supervision may induce sub-optimal reasoning strategies (Lightman et al., 2024).

2.2 MULTI-ROLE RL UNDER A SINGLE POLICY MODEL

Recently, studies have shown that a single policy model π_θ can effectively learn multiple roles by assigning role-specific rewards (Zhao et al., 2025; Liu et al., 2025b):

$$r^{\text{role}_i} \rightarrow \pi_\theta^{\text{role}_i}, \tag{3}$$

where r^{role_i} denotes the reward assigned to role_i, and $\pi_\theta^{\text{role}_i}$ represents the same underlying policy model π_θ instantiated under the prompt $\mathcal{P}^{\text{role}_i}$ corresponding to that role.

A typical application under this framework is to let the model simultaneously act as a reasoner and a data generator, achieving RL training starting from no human-labeled data (Zhao et al., 2025; Silver et al., 2017; Lu et al., 2025). More broadly, this multi-role, single-policy setup provides a natural substrate for self-evolving training: different roles instantiated from the same model can compete (Zhao et al., 2025; Liu et al., 2025a) or collaborate (Zhang et al., 2025b; Liu et al., 2025b), continuously improving the learning signals and behaviors (Tran et al., 2025). Building on this view, we investigate whether the same policy can simultaneously serve as a reasoner and a CoT rubrics generator (*i.e.,* rubricator). With role-specific rewards, the rubricator is encouraged to produce useful CoT supervision criteria, which in turn enables the self-evolution of CoT rewarding. In this case, the reasoner and rubricator collaborate together for better reasoning.

3 METHODOLOGY

In this section, we introduce **RLCER** (**R**einforcement **L**earning with **C**oT Supervision via Self-**E**volving **R**ubrics) for bootstrapping chain-of-thought reasoning. In the remainder of this section,

we first briefly introduce the key idea of RLCER in Section 3.1 to facilitate a high-level understanding. Then, we discuss the multiple roles under the same policy model in Section 3.2. After that, we discuss how to incentivize reasoning with self-proposed and rubrics in Section 3.3. We then discuss how to self-evolve such rubrics in Section 3.4. Finally, we present how to optimize the policy π_θ under the two roles based on role-specific rewards in Section 3.5.

3.1 Key Idea

The key idea of RLCER is to go beyond rewarding what LLMs answer (*i.e.,* final-answer correctness) to explicitly and autonomously reward how LLMs think (*i.e.,* CoT quality) using self-proposed, self-evolving rubrics. The rubrics are evolved by tracking how well their satisfaction correlates with the final answer correctness.

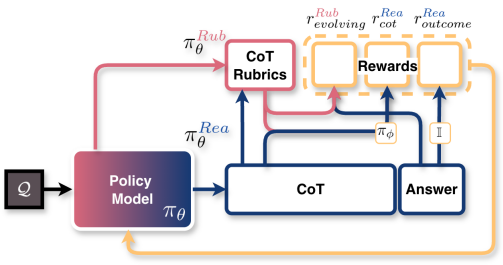


Figure 3: The RLCER loop. Format reward is ignored for brevity. One single policy model self-proposes rubrics for rewarding CoTs, and self-evolves the rubrics via rewarding generation capabilities.

To achieve this goal, we introduce two key roles in the multi-role RL framework for optimization: the reasoner π_θ^{Rea} for answering the question and the rubricator π_θ^{Rub} for generating rubrics supervising CoTs. Here, the reasoner $\pi_\theta^{Rea}(\cdot) = \pi_\theta(\cdot \mid \mathcal{P}^{Rea})$ and the rubricator $\pi_\theta^{Rub}(\cdot) = \pi_\theta(\cdot \mid \mathcal{P}^{Rub})$ share the same policy parameter θ, while their roles are distinguished by different prompts $\mathcal{P}^{Rea}$ and $\mathcal{P}^{Rub}$ respectively. The prompts can be found in Appendix D. Additionally, a verifier π_ϕ judges the satisfaction of rubrics, which is another fine-tuned and frozen model parameterized by ϕ.

As shown in Figure 3, we explicitly refine how the reasoner π_θ^{Rea} thinks, by rewarding the CoT quality beyond the answer correctness, with the CoT quality evaluated by the verifier π_ϕ based on rubrics generated by the rubricator. We further enable rubric generation to self-evolve for more autonomous and effective CoT supervision by rewarding the quality of generated rubrics, guided by the correlation between rubric satisfaction degree and answer correctness. We jointly optimize the policy model π_θ under two roles by assigning different rewards for each role. The advantages are computed separately for each role (Liu et al., 2025a).

3.2 Two Roles in One Policy: Reasoner and Rubricator

In this section, we take a closer look at the different roles of the reasoner π_θ^{Rea} and the rubricator π_θ^{Rub}, within the RLCER framework.

Reasoner π_θ^{Rea}. The reasoner answers the question by generating a CoT $\hat{\mathcal{C}}$ and a final answer $\hat{\mathcal{A}}$. Given the question $\mathcal{Q}$, the rollout sampling process of the reasoner can be formulated as:

$$\hat{\mathcal{C}}, \hat{\mathcal{A}} \sim \pi_\theta^{Rea}(\cdot \mid \mathcal{Q}), \tag{4}$$

where $\hat{\mathcal{C}}$ denotes the CoT process, and $\hat{\mathcal{A}}$ denotes the predicted final answer.

Rubricator π_θ^{Rub}. The rubricator generates a set of K rubrics for evaluating the CoT qualities based on the question $\mathcal{Q}$ and one CoT $\hat{\mathcal{C}}$. Given the question $\mathcal{Q}$ and a generated CoT $\hat{\mathcal{C}}$, the rollout sampling process of the rubricator can be formulated as:

$$\hat{\mathcal{R}} \sim \pi_\theta^{Rub}(\cdot \mid \mathcal{Q}, \hat{\mathcal{C}}), \qquad \hat{\mathcal{R}} \triangleq \{\hat{\tau}_k\}_{k=1}^{K},\ \hat{\tau}_k \triangleq (\hat{c}_k, \hat{s}_k). \tag{5}$$

The generated textual output $\hat{\mathcal{R}}$ contains K specific rubrics, each denoted as $\hat{\tau}_k$. Here, $\hat{c}_k$ is the textual criterion (*e.g.,* "avoids tangential explorations post-interval validation (3-∞) by focusing on count aggregation") of rubric $\hat{\tau}_k$ and $\hat{s}_k \in \mathbb{R}$ is the score of rubric $\hat{\tau}_k$ reflecting its importance, which can be both positive or negative. The rubrics generated under the rubricator role will then be used for judging the CoT quality through an external verifier π_ϕ.

3.3 Rewarding How to Think via Self-Proposed Rubrics

In this section, we discuss how to autonomously reward how LLMs think via self-proposed rubrics.

Rewarding CoT with self-proposed rubrics. Specifically, we reward a CoT based on how well it satisfies the valid rubrics proposed by the rubricator π_θ^{Rub}. The CoT reward is computed by aggregating the scores of the satisfied rubrics:

$$r_{cot}^{Rea} = \texttt{norm}\left(\sum_{\hat{\tau}_k \in \hat{\mathcal{R}}_{valid}} \pi_\phi\left(\hat{c}_k, \hat{\mathcal{C}}\right) \cdot \hat{s}_k\right), \tag{6}$$

where $\hat{\mathcal{R}}_{valid}$ denotes the set of valid rubrics from the rubricator, $\pi_\phi(\hat{c}_k, \hat{\mathcal{C}}) \in \{0, 1\}$ is a binary value produced by the verifier π_ϕ indicating whether the CoT $\hat{\mathcal{C}}$ satisfies rubric $\hat{c}_k$, and $\texttt{norm}(\cdot)$ is a normalization function that maps the aggregated score to the interval $[0, 1]$. The function $\texttt{norm}(\cdot)$ conducts the min–max normalization as $\texttt{norm}(x) = (x - \text{MinValue})/(\text{MaxValue} - \text{MinValue})$, where MinValue and MaxValue denote the minimum and maximum aggregated rubric scores attainable under the valid rubrics $\hat{\mathcal{R}}_{valid}$, respectively.

Valid rubrics. Here, a rubric $\hat{\tau}_k$ is considered valid if it is informative as a rewarding criterion: (i) its satisfaction indicator $\mathbf{v}_k$ is positively correlated with final-answer correctness $\mathbf{z}$ across N sampled rollouts of the same question $\mathcal{Q}$ (*i.e.,* $\texttt{corr}(\mathbf{v}_k, \mathbf{z}) > \alpha$), and (ii) it is discriminative across different CoTs and would not yield consistent satisfaction (*i.e.,* $\texttt{std}(\mathbf{v}_k) > 0$). We define $\mathbf{v}_k$ and $\mathbf{z}$ over these N rollouts as:

$$\begin{aligned} \mathbf{v}_k &\triangleq \left[\pi_\phi(\hat{c}_k, \hat{\mathcal{C}}_0), \ldots, \pi_\phi(\hat{c}_k, \hat{\mathcal{C}}_N)\right], \\ \mathbf{z} &\triangleq \left[\mathbb{I}(\mathcal{A}, \hat{\mathcal{A}}_0), \ldots, \mathbb{I}(\mathcal{A}, \hat{\mathcal{A}}_N)\right]. \end{aligned} \tag{7}$$

Intuitively, higher correlation implies that satisfying the rubric is consistently associated with correct final answers and can produce informative rewarding signals. We set $\alpha = 0.2$ by default.

Outcome reward. The CoT reward r_{cot}^{Rea} can be used as an auxiliary reward alongside the outcome reward $r_{outcome}^{Rea}$, which is calculated as follows:

$$r_{outcome}^{Rea} = \begin{cases} 1, & \texttt{is_equiv}(\mathcal{A}, \hat{\mathcal{A}}) \\ -1, & \text{Otherwise.} \end{cases} \tag{8}$$

Following DAPO (Yu et al., 2025a), we assign a binary outcome reward: $r_{outcome}^{Rea}$=1 if $\hat{\mathcal{A}}$ is equivalent to the ground truth $\mathcal{A}$, and $r_{outcome}^{Rea}$=-1 otherwise.

Total reward for reasoner. Finally, the overall reward for the reasoner is the combination of the outcome reward and the CoT reward:

$$r^{Rea} = r_{outcome}^{Rea} + r_{cot}^{Rea}. \tag{9}$$

Such a reward enables autonomously optimizing what LLMs answer and how LLMs think.

3.4 Rubrics Self-Evolving for Better Supervision

While the two-role design enables a single policy model to self-propose rubrics for CoT supervision, we argue that such rubrics may get saturated: once consistently satisfied by all rollouts, they lose discriminative power over CoT quality and no longer provide reward signals, as shown in Section 3.3. Therefore, we further propose to self-evolve the rubric generation capability, so as to progressively produce more informative and challenging rubrics that mitigate saturation.

Self-evolving with validity reward. To achieve rubricator self-evolving, we reward the fraction of the number of valid rubrics among all generated rubrics as follows:

$$r_{evolving}^{Rub} = \frac{K_{\text{valid}}}{K}, \tag{10}$$

where K and $K_{\text{valid}} \triangleq |\{\hat{\tau}_k^{valid}\}|$ are the numbers of all generated rubrics and valid rubrics, respectively. Such rewards encourage the rubricator to generate more informative rubrics that are positively

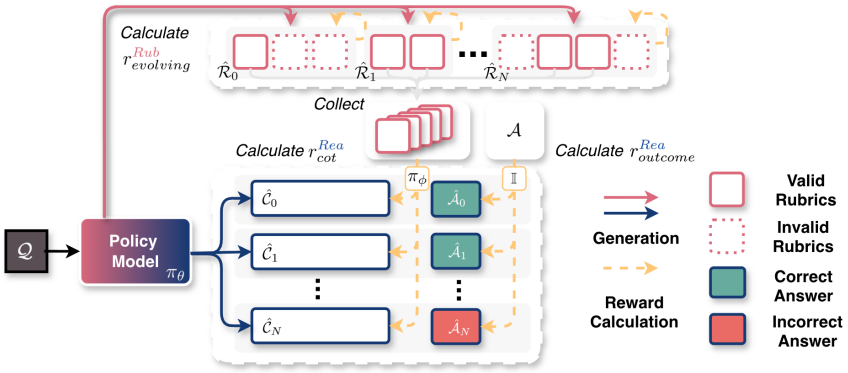


Figure 4: Illustration of the reward calculation process in RLCER. For question $\mathcal{Q}$, the reasoner generates N responses each with CoT $\hat{\mathcal{C}}_k$ and the final answer $\hat{\mathcal{A}}_k$. After that, the rubricator generates K_n specific rubrics (*i.e.,* $\hat{\mathcal{R}}_n \triangleq \{\hat{\tau}_k\}_{k=1}^{K_n}$). The outcome reward is applied first by matching the generated answer $\hat{\mathcal{A}}_k$ with the ground-truth answer $\mathcal{A}$. All the valid rubrics (*i.e.,* $\texttt{corr}(\mathbf{v}_k, \mathbf{z}) > \alpha$ and $\texttt{std}(\mathbf{v}_k) > 0$) are collected for rewarding CoTs. And the fraction of valid rubrics for the k-th rubricator generation (*i.e.,* $\frac{|\{\hat{\tau}^{valid}_{n,k}\}|}{K_n}$) rewards the rubricator for self-evolving. correlated with answer correctness (*i.e.,* $\texttt{corr}(\mathbf{v}_k, \mathbf{z}) > \alpha$) and discriminative (*i.e.,* $\texttt{std}(\mathbf{v}_k) > 0$), thereby preventing rubric saturation. This reward provides a simple way for evolving the quality of generated rubrics, by connecting the satisfaction of the rubrics with the final answer correctness.

Format reward. Additionally, to ensure the proposed rubrics can be stably parsed, we reward the rubricator with a format reward, rewarding the correct format as follows:

$$r^{Rub}_{format} = \begin{cases} 1, & \text{The format is correct.} \\ 0, & \text{Otherwise.} \end{cases} \tag{11}$$

Total reward for rubricator. Finally, the total reward for the rubricator is calculated as the combination of the quality reward and the format reward:

$$r^{Rub} = r^{Rub}_{evolving} + r^{Rub}_{format}. \tag{12}$$

Such a reward design encourages the rubricator to continuously refine its proposed rubrics, consistently providing informative rewarding signals for CoT supervision.

3.5 Two-Role Optimization under a Single Policy

After computing the rewards for the reasoner and the rubricator, we jointly optimize the shared policy model π_θ under both roles in an end-to-end manner. Specifically, we compute role-specific advantages $\hat{A}^{Rea}_t$ and $\hat{A}^{Rub}_t$ from their respective rewards r^{Rea} and r^{Rub}, and use them to update the policy via a standard policy-gradient objective. Gradients from both roles are aggregated to update the same set of parameters θ, allowing the model to learn from role-specific feedback while maintaining a unified policy. The overall optimization procedure is illustrated in Algorithm 1.

With role-specific advantages $\hat{A}^{Rea}_t$ and $\hat{A}^{Rub}_t$, the optimization objective can be formulated as:

$$\begin{aligned} \mathcal{J}(\theta) = \; & \mathbb{E}_{(\mathcal{Q},\mathcal{A})\sim\mathcal{D}^{Rea},\, o\sim\pi^{Rea}_{\theta_{\text{old}}}(\cdot|\mathcal{Q})}\Big[\min\Big(\rho_t(\theta)\,\hat{A}^{Rea}_t,\; \text{clip}\big(\rho_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\hat{A}^{Rea}_t\Big)\Big], \\ & + \mathbb{E}_{(\mathcal{Q},\hat{\mathcal{C}})\sim\mathcal{D}^{Rub},\, o\sim\pi^{Rub}_{\theta_{\text{old}}}(\cdot|\mathcal{Q},\hat{\mathcal{C}})}\Big[\min\Big(\rho_t(\theta)\,\hat{A}^{Rub}_t,\; \text{clip}\big(\rho_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\hat{A}^{Rub}_t\Big)\Big], \end{aligned} \tag{13}$$

where $\rho_t(\theta)$ is the policy likelihood ratio between π_θ and $\pi_{\theta_{\text{old}}}$ at step t, and ϵ is the clipping hyperparameter (*e.g.,* 0.2). It is worth noting that GRPO is unsuitable for our algorithm, since the rubricator operates under different contexts, preventing its rollouts from being grouped under the same context required by GRPO (Shao et al., 2024).

4 Experiments

- **RQ1:** Can self-proposed rubrics provide reliable rewarding signals for RL training?

Algorithm 1: Policy Optimization with RLCER

Input: Question $\mathcal{Q}$, answer $\mathcal{A}$, rollout sampling number N, and maximum training steps T.
Initialization: initialize policy model π_θ, training step $t \leftarrow 0$.
while $t \leq T$ **do**
  // sample reasoner rollouts
  $\{\hat{\mathcal{C}}_n, \hat{\mathcal{A}}_n\}_{n=1}^{N} \sim \pi_\theta^{Rea}(\cdot \mid \mathcal{Q})$
  // sample rubricator rollouts
  $\{\hat{\mathcal{R}}_n\}_{n=1}^{N} \sim \pi_\theta^{Rub}(\cdot \mid \mathcal{Q}, \hat{\mathcal{C}_n})$ **Role-specific rollout**
  // calculate reasoner rewards
  $\{r_n^{Rea}\}_{n=1}^{N} = \{r_{outcome,n}^{Rea} + r_{cot,n}^{Rea}\}_{n=1}^{N}$
  // calculate rubricator rewards
  $\{r_n^{Rub}\}_{n=1}^{N} = \{r_{evolving,n}^{Rub} + r_{format,n}^{Rub}\}_{n=1}^{N}$ **Rewarding**
  // calculate role-specifc advantages
  Calculate $\hat{A}_t^{Rea}$ and $\hat{A}_t^{Rub}$
  // policy optimization
  Update policy π_θ via $\mathcal{J}(\theta)$ **Policy updating**
  $t \leftarrow t + 1$
end while
return π_θ.

- **RQ2:** How does RLCER enhance the reasoning capabilities of LRMs?
- **RQ3:** How does the self-evolving enable the rubricator to gradually propose better rubrics?
- **RQ4:** How effective are generated rubrics at facilitating reasoning when used as in-prompt hints?

4.1 Experiment Setup

We briefly introduce the experiment setup here; more details can be found in Appendix B.

LLMs. We conduct experiments on two open-source Qwen models of different sizes (8B and 4B) (Yang et al., 2025). We first cold-start the pre-trained models (*i.e.,* Qwen3-8B-Base and Qwen3-4B-Base) by performing supervised fine-tuning on correctly formatted data obtained via reject sampling from a stronger teacher model, as we observe that even post-trained versions of such small-sized models (*e.g.,* Qwen3-8B) cannot always strictly follow the required instructions when acting as the rubricator and reasoner. More details can be found in Appendix B. All subsequent RL experiments are conducted based on these carefully cold-started models.

Training and evaluation. We train models based on the DAPO-Math-17k dataset (Yu et al., 2025a), which contains 17k high-quality math questions. We evaluate models on various reasoning benchmarks. For math reasoning, we evaluate math-reasoning benchmarks including AIME24 (Mathematical Association of America, 2024), AIME25 (Mathematical Association of America, 2024), and AMC23 (AI-MO, 2023). For general knowledge reasoning, we evaluate on GPQA-Diamond (Rein et al., 2023) and SuperGPQA (M-A-P Team, 2025). For the large size of SuperGPQA, we evaluate on three subsets, each including 100 questions, namely SuperGPQA-Eng, SuperGPQA-Med, and SuperGPQA-Sci. For each testing question, we sample N=16 responses with sampling temperature as 0.7. We report pass@1 as the accuracy metric (*i.e.,* the average pass rate among 16 responses). For all RL training, we fix the max training steps to 1500.

4.2 (RQ1) Reliability of Self-Proposed Rubrics

A key factor that determines the success of our method lies in whether the self-proposed and self-evolving rubrics can provide meaningful learning signals for RL. To verify this, we first conduct a preliminary experiment in an outcome-free setting that trains the model by **only rewarding CoTs based on self-proposed rubrics without any outcome rewards** (*i.e.,* $r^{Rea} = r_{cot}^{Rea}$).

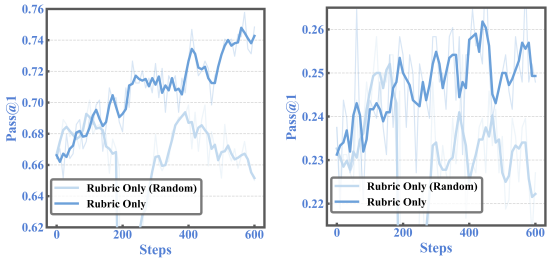
If the model still learns to reason effectively under this setup, it would indicate that self-proposed rubrics can yield meaningful RL signals. We report the accuracy dynamics on reasoning tasks during the training in Figure 5, where the bold lines are the rolling averaged performance with a

Table 1: The performance comparison across diverse reasoning benchmarks.

Scale	Method	Datasets						
		AIME2024	AIME2025	AMC2023	GPQA-Diamond	SuperGPQA-Eng	SuperGPQA-Med	SuperGPQA-Sci
4B	Base	11.25	6.46	31.09	7.77	18.50	14.25	15.44
	SFT	17.29	18.96	59.53	24.43	31.69	28.25	27.81
	+ RLVR	29.38	30.21	79.53	44.16	39.75	28.50	**42.88**
	+ RLCER	**29.38**	**30.63**	**81.88**	**44.91**	**40.19**	**31.63**	41.81
8B	Base	11.46	9.58	42.34	24.37	33.06	24.69	23.63
	SFT	22.29	23.75	66.41	31.72	36.00	33.75	35.19
	+ RLVR	34.79	32.50	84.53	46.56	42.94	**38.31**	48.81
	+ RLCER	**37.50**	**33.33**	**86.41**	**48.77**	**45.00**	36.50	**50.25**

window size of 3 and the slim and shallow lines are the original performance. We have the following observation:

Self-proposed rubrics can provide meaningful CoT rewarding signals for RL training. As shown in this figure, by only rewarding CoTs with the self-proposed rubrics (*i.e.,* Rubric Only in the figure), the reasoning capability of the LLM shows consistent improvement during training. In contrast, when we assign random values between 0 and 1 as rubric rewards (*i.e.,* Rubric Only (Random) in the figure), the LLM fails to achieve performance improvements and even exhibits sudden performance drops during training around 200 steps. Such results show that the self-proposed and self-evolving rubrics can yield meaningful CoT rewarding signals, which lay an important foundation for our approach.



(a) Accuracy on AMC23. (b) Accuracy on AIME25.

Figure 5: Accuracy dynamics when only rewarding CoTs. Even when rewarding CoTs with self-proposed rubrics without any outcome reward can bring consistent performance gain.

4.3 (RQ2) Effectiveness of RLCER

Based on the reliability of self-proposed rubrics, we further test the effectiveness of combining both $r^{Rea}_{outcome}$ and r^{Rea}_{cot} in RLCER. We compare RLCER with the vanilla RLVR which only considers the outcome reward, and report the performance comparison in Table 1. Here, both RLCER and RLVR start from the SFT checkpoint. We have the following observations:

RLCER outperforms the outcome-centric RLVR training. As shown in Table 1, training with RLCER outperforms the vanilla RLVR across multiple datasets and model sizes on average. Even when trained only on math datasets, RLCER also generalizes well on general reasoning tasks (*i.e.,* GPQA series.) Additionally, we observe that the performance gain is more obvious on larger size models (*i.e.,* 8B models benefit more from RLCER than 4B models). Such phenomena validate the effectiveness of RLCER, which provides a "free-lunch" for improving the RL performance, by enhancing it with self-proposed and self-evolving rubrics. Case studies of the generated rubrics are in Appendix E.

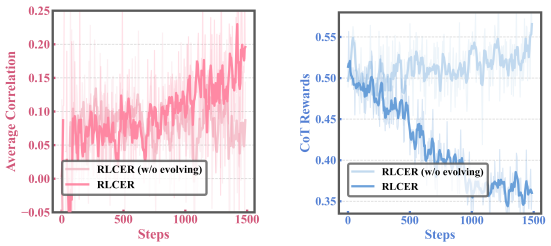
Statistical significance. We conduct paired comparisons between RLCER and RLVR on Table 1 at matched model scales and benchmark columns; the improvements aggregate to statistically significant gains under standard two-sided tests at significance level $p < 0.05$.

4.4 (RQ3) Mechanism of Rubrics Self-Evolving

In this section, we study how the rubrics self-evolving affects training within the RLCER framework (*i.e.,* the role of the reward $r^{Rub}_{evolving}$). To study this, we conduct an ablation study on the 8B-sized model by abandoning the evolving reward $r^{Rub}_{evolving}$, and the rubricator is only rewarded with the format reward. We report the training dynamics in Figure 1 and Figure 6 including the averaged performance (*i.e.,* averaged accuracy among AIME2024, AIME2025, and AMC2023), the averaged correlation between rubrics satisfaction and final answer correctness (*i.e.,* $\texttt{corr}(\mathbf{v}_k, \mathbf{z})$), and the CoT reward of the reasoner (*i.e.,* r^{Rea}_{cot}). The performance is reported as the rolling average of 5 scores

for a clearer demonstration, where the bold lines report the rolling average scores and the shallow and slim lines report the raw scores. We have the following observations:

Self-evolving rubrics enable better performance. As shown in Figure 1, by rewarding with $r^{Rub}_{evolving}$, RLCER exhibits a more stable learning curve and gradually outperforms the ablation version (*i.e.,* RLCER (w/o evolving)). Additionally, both RLCER and RLCER (w/o evolving) outperform the naive RLVR method as it quickly converges to a sub-optimal performance. This shows that self-evolving enhances reasoning.



(a) Correlation dynamics. (b) r^{Rea}_{cot} dynamics.

Figure 6: The effect of self-evolving on correlation and CoT-reward dynamics.

Self-evolving with $r^{Rub}_{evolving}$ enables more informative rubrics. We further analyze whether self-evolving enables the policy model to propose better rubrics. As shown in Figure 6a, the averaged correlation between rubric satisfaction and final answer correctness (*i.e.,* $\texttt{corr}(\mathbf{v}_k, \mathbf{z})$) increases along the training process, while the ablation baseline shows an unchanged correlation. This indicates that self-evolving enables the policy model to gradually propose more informative rubrics that better align with the final answer correctness. Additionally, with such evolved rubrics, it gradually becomes harder to gain the CoT reward $r^{Rub}_{evolving}$ as the Figure 6b shows a decreasing trend during the RLCER training. While the ablation baseline's rubrics can be gradually satisfied by π^{Rea}_{θ} as $r^{Rub}_{evolving}$ increases. Such results highlight the importance of rubrics self-evolving.

4.5 (RQ4) Effectiveness of Rubrics as In-prompt Reasoning Hints

We aim to understand how generated rubrics guide models toward better reasoning. While we have confirmed that self-proposed and self-evolved rubrics can provide valid reward signals for CoTs, their mechanism of implicitly guiding reasoning remains unclear due to the difficulty in analyzing underlying reward signals. To address this, we use these rubrics as in-prompt hints and evaluate the model's reasoning performance. We report the performance of Qwen3-8B (with a max response length of 20480) in Figure 7, which shows improved performance when rubrics are used as hints—proving the rubrics effectively guide reasoning. Additionally, BoN (N=16) achieves further improvement on the AIME dataset, indicating rubric hints may encourage more exploration.

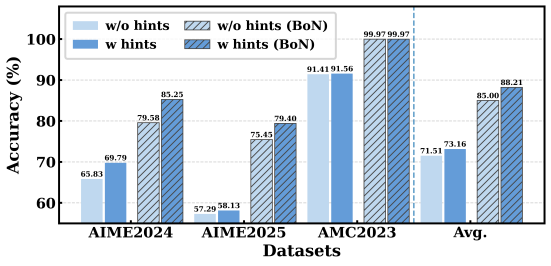


Figure 7: Rubrics as in-prompt hints performance.

5 Conclusion

Outcome-centric RLVR improves answer accuracy but neglects direct CoT supervision, which may lead to suboptimal reasoning. Directly rewarding CoTs is hard, as reward models are costly to train and vulnerable to distribution shift and reward hacking. We propose RLCER to reward CoTs via self-built, self-evolving rubrics without human annotation. It consistently outperforms outcome-only RLVR across datasets and LLMs, verifying such rubrics provide effective CoT supervision to boost LLM reasoning performance. We will further explore its generalization to non-verifiable domains.

Limitations. There are still several limitations in this paper. On the one hand, the introduction of the rubricator role increases the rollout burden and thereby requires more training time. On the other hand, our method is still quite limited to the RLVR domain, leaving the effectiveness of rewarding with self-proposed rubrics on non-verifiable domains unknown.

AI USE STATEMENT

We used generative AI tools to improve the readability of this paper and to format the manuscript and its references. All such edits were reviewed and verified by the authors. The cold-start and verifier training data are generated by an LLM, which is a component of the proposed method described in Appendix B rather than an authoring aid. Beyond the above, we did not use generative AI tools for developing the conceptual framework, formulating hypotheses, designing the methodology or experiments, or interpreting the results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work does not involve human subjects, crowdsourcing, or the collection of personal data. All base models, datasets, benchmarks, and software frameworks we use are publicly available and are cited in the paper; their licenses are summarized in Appendix G. Stronger reasoning models could lower the cost of misuse (*e.g.*, automated cheating on graded assessments or persuasive step-by-step misinformation), and rubrics learned by the policy may inherit biases from prompts, rollouts, or the verifier. We discuss these risks, together with the expected benefits and possible mitigations, in Appendix F.

REPRODUCIBILITY STATEMENT

Section 3.3 and Section 3.4 define every reward term (outcome, CoT, validity, and format rewards) together with the rubric-validity criterion, and Section 3.5 and Algorithm 1 specify the two-role optimization procedure. The experimental setup (base models, training data, benchmarks, and evaluation protocol) is described in Section 4.1, and Appendix B lists the cold-start SFT and RL hyperparameters (Tables 2 and 3). The exact prompts of the reasoner, the rubricator, and the verifier are given in Appendix D, and a worked example of generated rubrics is given in Appendix E. All base models (Qwen3-4B/8B-Base), the training set (DAPO-Math-17k), the benchmarks, and the training frameworks (LlamaFactory and verl) are publicly available (Appendix G). Our code and the teacher-generated cold-start data cannot be released at submission time due to our organization's policy.

REFERENCES

AI-MO. American mathematics contest. https://huggingface.co/datasets/AI-MO/aimo-validation-amc, 2023.

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. *CoRR*, abs/2505.08775, 2025.

Edward Y. Chang, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *CoRR*, abs/2502.03373, 2025.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *ICML*. OpenReview.net, 2024.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. *CoRR*, abs/2505.11831, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *CoRR*, abs/2502.01456, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, 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, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, 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, 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, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kai-Xuan Chen, Mingli Song, and Dacheng Tao. Serl: Self-play reinforcement learning for large language models with limited data. *CoRR*, abs/2505.20347, 2025.

Yunzhen Feng, Julia Kempe, Cheng Zhang, Parag Jain, and Anthony Hartshorn. What characterizes effective reasoning? revisiting length, review, and structure of cot. *CoRR*, abs/2509.19284, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: On path to artificial super intelligence. *CoRR*, abs/2507.21046, 2025.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *CoRR*, abs/2507.17746, 2025.

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. *CoRR*, abs/2508.12790, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, and Ilge Akkaya. Openai o1 system card. *CoRR*, abs/2412.16720, 2024.

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 *ICLR*. OpenReview.net, 2025.

Gangwei Jiang, Yahui Liu, Zhaoyi Li, Qi Wang, Fuzheng Zhang, Linqi Song, Ying Wei, and Defu Lian. What makes a good reasoning chain? uncovering structural patterns in long chain-of-thought reasoning. *CoRR*, abs/2505.22148, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. SPICE: self-play in corpus environments improves reasoning. *CoRR*, abs/2510.24684, 2025a.

Mickel Liu, Liwei Jiang, Yancheng Liang, Simon Shaolei Du, Yejin Choi, Tim Althoff, and Natasha Jaques. Chasing moving targets with online self-play reinforcement learning for safer language models. *CoRR*, abs/2506.07468, 2025b.

Hongliang Lu, Yuhang Wen, Pengyu Cheng, Ruijin Ding, Haotian Xu, Jiaqi Guo, Chutian Wang, Haonan Chen, Xiaoxi Jiang, and Guanjun Jiang. Search self-play: Pushing the frontier of agent capability without supervision. *CoRR*, abs/2510.18821, 2025.

M-A-P Team. Supergpqa: Scaling LLM evaluation across 285 graduate disciplines. *CoRR*, abs/2502.14739, 2025.

Mathematical Association of America. American invitational mathematics examination (aime), February 2024. URL https://maa.org/math-competitions/american-invitational-mathematics-examination-aime.

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. *CoRR*, abs/2311.12022, 2023.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815, 2017.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *CoRR*, abs/2501.06322, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Junkang Wu, Kexin Huang, Jiancan Wu, An Zhang, Xiang Wang, and Xiangnan He. Quantile advantage estimation: Stabilizing RLVR for LLM reasoning. *CoRR*, abs/2509.22611, 2025.

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, Jian Yang, Jiaxi Yang, 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. *CoRR*, abs/2505.09388, 2025.

Zhiling Ye, Yun Yue, Haowen Wang, Xudong Han, Jiadi Jiang, Cheng Wei, Lei Fan, Jiaxin Liang, Shuowen Zhang, Ji Li, Chunxiao Guo, Jian Wang, Peng Wei, and Jinjie Gu. Self-rewarding rubric-based reinforcement learning for open-ended reasoning. *CoRR*, abs/2509.25534, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In *NeurIPS*, 2025a.

Tianyu Yu, Bo Ji, Shouli Wang, Shu Yao, Zefan Wang, Ganqu Cui, Lifan Yuan, Ning Ding, Yuan Yao, Zhiyuan Liu, Maosong Sun, and Tat-Seng Chua. RLPR: Extrapolating RLVR to general domains without verifiers. *CoRR*, abs/2506.18254, 2025b.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *ICML*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*, 2022.

Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and rl on reasoning language models. *arXiv preprint arXiv:2512.07783*, 2025a.

Jingyu Zhang, Haozhu Wang, Eric Michael Smith, Sid Wang, Amr Sharaf, Mahesh Pasupuleti, Benjamin Van Durme, Daniel Khashabi, Jason Weston, and Hongyuan Zhan. The alignment waltz: Jointly training agents to collaborate for safety. *CoRR*, abs/2510.08240, 2025b.

Junkai Zhang, Zihao Wang, Lin Gui, Swarnashree Mysore Sathyendra, Jaehwan Jeong, Victor Veitch, Wei Wang, Yunzhong He, Bing Liu, and Lifeng Jin. Chasing the tail: Effective rubric-based reward modeling for large language model post-training. *CoRR*, abs/2509.21500, 2025c.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, Yu Fu, Xingtai Lv, Yuchen Zhang, Sihang Zeng, Shang Qu, Haozhan Li, Shijie Wang, Yuru Wang, Xinwei Long, Fangfu Liu, Xiang Xu, Jiaze Ma, Xuekai Zhu, Ermo Hua, Yihao Liu, Zonglin Li, Huayu Chen, Xiaoye Qu, Yafu Li, Weize Chen, Zhenzhao Yuan, Junqi Gao, Dong Li, Zhiyuan Ma, Ganqu Cui, Zhiyuan Liu, Biqing Qi, Ning Ding, and Bowen Zhou. A survey of reinforcement learning for large reasoning models. *CoRR*, abs/2509.08827, 2025d.

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

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *ACL*, 2025.

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

A Related Works

A.1 LLM Reasoning with RL

Reinforcement learning has become an essential technique for incentivizing chain-of-thought (CoT) reasoning in LLMs (Zhang et al., 2025d; Zhao et al., 2023). A milestone paradigm is reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; Jaech et al., 2024; DeepSeek-AI et al., 2025), where LLMs are trained on verifiable tasks such as math problems, puzzle solving (Chollet et al., 2025), and code execution (Jain et al., 2025), by rewarding final-answer correctness using task-specific verifiers. Since such tasks are intrinsically difficult to reward-hack, RLVR tends to encourage LLMs to produce longer CoTs before answering, thereby improving reasoning performance (DeepSeek-AI et al., 2025; Jaech et al., 2024). However, despite its effectiveness in incentivizing reasoning, most RLVR training remains outcome-centric and provides no direct supervision over the CoT itself, lacking guidance on how the model should reason. While several efforts have tried to mitigate this by assigning process rewards using additional reward models (RM) (Lightman et al., 2024), training such RMs requires labor-intensive training data collection (Lightman et al., 2024; Cui et al., 2025; Zheng et al., 2025), and a static RM often fails to keep pace with the evolving CoT distribution during training (DeepSeek-AI et al., 2025). Therefore, such challenges hinder the combination of RLVR training with CoT supervision criteria for rewarding in practice.

A.2 Self-Evolving in LLMs

A critical step toward general artificial intelligence lies in the ability to self-evolve with little or no human intervention (Gao et al., 2025). Recent work has made progress via self-training and self-play, where a single policy model π_θ can self-generate training data (Zhao et al., 2025; Lu et al., 2025; Liu et al., 2025a), or self-provide reward signals during learning (Chen et al., 2024; Fang et al., 2025; Yuan et al., 2024). The self-evolving capabilities can further be strengthened with techniques such as multi-role reinforcement learning, where a single policy model π_θ is prompted to play different roles (*e.g.,* reasoner, data generator, or reward provider) and trained with role-specific rewards (Zhao et al., 2025; Liu et al., 2025a; Zhang et al., 2025b). Therefore, these advances in such self-evolving methods suggest the possibilities of self-proposing criteria for CoT supervision, while this direction remains largely under-explored.

A.3 Reinforcement Learning with Rubrics

Recently, reinforcement learning with rubrics has emerged as a promising way to provide fine-grained and interpretable reward signals (Huang et al., 2025; Gunjal et al., 2025; Arora et al., 2025; Zhang et al., 2025c). Here, rubrics are structured, checkable evaluation criteria (*i.e.,* explicit requirements for assessing outputs) (Gunjal et al., 2025) that decompose high-level objectives into multi-facet supervision signals (Shao et al., 2025), enabling RL to extend towards non-verifiable and open-ended domains (Zhou et al., 2025; Gunjal et al., 2025). However, existing work largely applies rubrics to score response quality, leaving their effectiveness for CoT supervision underexplored. Moreover, most approaches rely on predefined or static rubrics, which may become mismatched as model behaviors evolve during training (Shao et al., 2025). To address these gaps, we introduce a framework that self-proposes rubrics as CoT supervision criteria and continually evolves them throughout training, integrating rubric-based RL for CoT supervision with self-evolving training.

B Implementation Details

We conduct the cold-start training with the LlamaFactory [1] and all the RL experiments based on the verl [2] framework.

Compute disclosure. Our organization does not permit public disclosure of GPU/TPU hardware models, cluster topology, device counts, memory envelopes, wall-clock duration, or aggregate GPU-days for the experiments reported in this paper; we therefore omit concrete compute specifications beyond software and hyperparameters below.

Cold-start. For the cold-start stage, we collect 40k training data from the responses of a proprietary teacher model (accessed via an API), including 20k math reasoning trajectories and 20k rubricator trajectories, which are collected via reject sampling. We then perform full-parameter supervised fine-tuning (SFT) on the 40k dataset using Qwen3 prompt template, with a sequence cutoff length of 32768, per-device train batch size = 1, gradient accumulation steps = 1, base learning rate = 2e-5 (cosine scheduler with min lr = 2e-6), weight decay = 0.1, warmup ratio = 0.01 and total training epochs = 5.0. For the verifier, we distill a small-sized model Qwen3-4B-Base based on the verification responses from the same proprietary teacher model, using the same SFT recipe as in the cold-start stage.

Table 2: Key training hyperparameters used in the cold-start SFT.

Hyperparameter	Value
Prompt template	Qwen3
Sequence cutoff length	32768
Per-device train batch size	1
Gradient accumulation steps	1
Learning rate (cosine scheduler)	2e-5 (min lr = 2e-6)
Weight decay	0.1
Warmup ratio	0.01
Epochs	5.0

[1] https://github.com/hiyouga/LlamaFactory
[2] https://github.com/volcengine/verl

RL training. For the RL training, we train all the models with the DAPO-Math-17K data, with prompt max length = 16384, response max length = 12288 (overlong buffer enabled, length = 4096), no KL reward/loss, clip ratio = 0.2, learning rate for actor = 1e-6, learning rate for critic = 1e-5, temperature = 1.0, top_p = 1.0, train batch size = 32, rollout number N = 8, mini batch size = 64, weight decay=0.1, grad clip=1.0, lr warmup steps=10. For all RL training, we fix the max training steps to 1500. The response length and prompt length are configured in this setting because the outputs generated by the reasoner will serve as the input prompts for the rubricator. Therefore, the max prompt length should be greater than the response length, and the sum of these two values must not exceed 32k tokens.

Table 3: Key training hyperparameters used in RL training.

Hyperparameter	**Value**
Algorithm	PPO
Training dataset	DAPO-Math-17K
Max prompt length	16384
Max response length	12288
Overlong buffer	4096
KL reward/loss	None
Clip ratio	0.2
Learning rate (actor)	1e-6
Learning rate (critic)	1e-5
Sampling temperature	1.0
Top_p	1.0
Train batch size	32
Rollout number (N)	8
Mini batch size	64
Weight decay	0.1
Gradient clip	1.0
LR warmup steps	10
Max steps	1500

C Additional Experiments

Evaluation stability across random seeds. The results reported in the main paper are the average accuracy over N=16 sampling runs, which is a standard practice in LLM evaluation (Yu et al., 2025b; Wu et al., 2025). With the sampling temperature set to 0.7, such reported results can be regarded as samples drawn under 16 different random conditions. To further verify the evaluation stability, we re-run the evaluation of the 8B model with 3 different random seeds, with 16 independent samples per run. We report the mean and the standard deviation of the three runs in Table 4, which show that our evaluation results are stable with small variance.

Table 4: Evaluation stability of RLCER (8B) across 3 random seeds, with N=16 samples per run.

Method	AIME2024	AIME2025	AMC2023	GPQA-Diamond	SuperGPQA-Eng	SuperGPQA-Med	SuperGPQA-Sci
RLCER	37.47 ± 4.5	33.25 ± 3.6	86.42 ± 1.5	48.86 ± 1.6	44.89 ± 5.6	36.62 ± 1.4	50.33 ± 3.2

Sensitivity to the correlation threshold α. We study the sensitivity of RLCER to the correlation threshold α that determines rubric validity, by training the 8B model with $\alpha = 0$ and $\alpha = 0.5$ besides the default $\alpha = 0.2$. As shown in Table 5, a moderate value of α yields relatively better results, while RLCER consistently outperforms the naive RLVR (*cf.* Table 1) under all the α settings.

Table 5: Sensitivity of RLCER (8B) to the correlation threshold α.

α	AIME2024	AIME2025	AMC2023
0	35.63	32.29	84.53
0.2	37.50	33.33	86.41
0.5	36.67	32.91	86.09

Comparison with frozen and externally generated rubrics. To validate the necessity of the self-proposing and self-evolving design, we compare RLCER with two additional rubric-based RL baselines on the 8B model. For the *Frozen* setting, the model generates rubrics based on the question and the initial CoT solution, and these rubrics remain frozen throughout the RL process. For the *External* setting, we directly employ an external large model (*i.e.,* Seed 1.6 Pro) as the rubricator. We report the results in Table 6. As shown, frozen rubrics underperform the dynamic RLCER approach, while using the external large model as the rubricator only achieves comparable performance to RLCER. It is worth noting that the external large model has far more parameters than the policy model. Therefore, RLCER's strong performance on such small-scale models demonstrates the effectiveness of our approach.

Table 6: Comparison with frozen rubrics and rubrics generated by an external large model (8B).

Method	AIME2024	AIME2025	AMC2023
RLVR	34.79	32.50	84.53
Frozen	35.63	33.13	83.75
External	38.58	32.50	86.09
RLCER	37.50	33.33	86.41

Faithfulness of the distilled verifier. As the verifier is distilled from a proprietary teacher model into a small-sized model (*i.e.,* Qwen3-4B-Base), we further evaluate its verification quality. We select 1,000 CoT samples and ask both the teacher model and the distilled small model to assess them using the same rubrics, and then measure the proportion of cases where their judgements are exactly consistent. The agreement rate reaches 91.6%, which demonstrates the faithfulness of our distilled verifier.

Do the rubrics reflect genuine reasoning quality? Since rubric validity is determined by the correlation with final-answer correctness, one may concern that the proposed rubrics capture answer-correlated shortcuts rather than genuine reasoning quality. Besides the case study in Appendix E and the in-prompt hints experiment in Section 4.5, we further randomly sample 200 generated rubrics and use an external model (*i.e.,* Seed 2.0 Pro) to score on a 0–10 scale whether they reflect authentic

reasoning rather than relying on shortcuts, where 10 means the rubrics reflect the real reasoning gap and 0 means the rubric is leading a shortcut. The 200 samples achieve an average score of 8.65. Moreover, if the proposed rubrics were incentivizing the model to take shortcuts, we would not expect to observe the generalization to general reasoning tasks (*e.g.,* the GPQA series in Table 1) when the model is trained solely on mathematical tasks. These results confirm that the rubrics proposed by the model effectively guide high-quality reasoning.

D PROMPTS

In this section, we report the prompts used as the reasoner π_θ^{Rea} and the rubricator π_θ^{Rub}. It is worth mentioning that during our initial attempts to directly prompt the proprietary teacher model to generate rubrics based on the question and a reference answer, the model was unable to generate rubrics that are not already presented in the reference. Therefore, we modified the prompt to instruct the model to generate a set of rubrics where the current response score falls below the middle value, and explicitly reinforced this in the format reward. We observed that this practice triggers the model to actively refine rubrics during reasoning to generate meaningful rubrics for room of self-improvement. The 20k rubricator trajectories were then collected using this specific prompt with rejection sampling under such format reward.

PROMPT:

```
{question}
Please reason step by step, and put your final answer within \boxed{}.
```

Figure 8: The prompt of the reasoner role.

PROMPT:

```
You are an expert in educational assessment and rubric design. Your task is to analyze
a given question-answer pair and generate comprehensive evaluation rubrics that can be
used to assess response quality for this question. The answer is only a reference
answer from the student(not necessarily a good response), so your rubric system should
consider the merits already presented in the response, and most importantly, room for
further improvements.
# Input Data
[Question]: {question}
[Response]: {response}
# Task Instructions
Based on the provided question and answer, generate a comprehensive rubric that's
suitable with multiple evaluation criteria. Each criterion should be:
1. Specific and Measurable: Clearly define what constitutes meeting or not meeting
the criterion
2. Binary Evaluable: Can be assessed as true/false by an LLM evaluator
3. Comprehensive Coverage: Together, all criteria should cover the key aspects of a
high-quality response
# Required Rubric Categories
Generate criteria to cover these aspects: 1. Effectiveness of Problem Decomposition &
Planning Definition: The systematic breakdown of a complex problem into logically
ordered, manageable sub-tasks, and the formulation of a strategic roadmap for solving
them. Core Aspects: Identifying key components, establishing step dependencies,
sequencing operations, and anticipating challenges before execution.
2. Effectiveness of BackTracking / Self-Validation / Error Handling Definition: The
continuous monitoring of reasoning validity through consistency checks, proactive error
detection, and adaptive revision of flawed steps. Core Aspects: Implementing
verification checkpoints (e.g., unit analysis/estimation), diagnosing inconsistencies,
recovering from dead ends via path correction (not restarting), and mitigating error
propagation.
3. Reasoning Clarity & Flow Definition: The coherent structuring and communication of
logical progression where each step explicitly follows from prior conclusions and leads
to subsequent insights. Core Aspects: Hierarchical argument organization under clear
objective, unambiguous terminology, explicit logical connectors (e.g., "thus," "since,"
"conversely"), and seamless transitions between ideas.
4. Reasoning Focus & Efficiency Definition: The maintenance of strict alignment with
problem objectives while optimizing cognitive resources through elimination of
redundancies and irrelevant explorations. Core Aspects: Sustained direction toward
core goals, pruning of tangential paths, and avoidance of overthinking/repetition
during reasoning.
5. Other Question-Specific Aspects Definition: The strategic selection and
application of domain-relevant methods that align with both problem constraints and
solution goals. Core Aspects: Leveraging context-optimal approaches (e.g.,
dimensional analysis in physics, elimination methods for multiple-choice), avoiding
technique mismatches, and adapting tools to exploit problem structure (e.g., symmetry
in geometry).
```

Figure 9: The prompt of the rubricator role.

```
# Output Format Return a JSON object with the following structure:
"'json
{
  "question_domain":  "math/calculus",
  "rubrics":  [ { "category":  "Effectiveness of Problem Decomposition & Planning"
"criterion":  "XXX", "points":  5 }, { "category":  "Reasoning Clarity & Flow"
"criterion":  "XXX", "points":  3 }, { "category":  "Reasoning Focus & Efficiency"
"criterion":  "XXX", "points":  4 }, { "category":  "Other Question-Specific Aspects"
"criterion":  "XXX", "points":  -3 }, { "category":  "Reasoning Focus & Efficiency"
"criterion":  "XXX", "points":  -2 } // ...  additional criteria ],
  "maximum_score":  a(sum of all positive metrics),
  "minimum_score":  b(sum of all negative metrics),
  "current_score":  x(must be smaller than (a+b)/2, otherwise revise your rubric system
to get better room of improvement)
}
"'
# Important Guidelines - First output the domain of the question in two levels, then
generate 5 - 15 criteria in total, ensuring comprehensive coverage - Points should
reflect the relative importance of each criterion (supports positive scores from 1 to 5
for reward criteria, and negative scores from -5 to -1 for penalty criteria).
Specifically, a penalty criteria should be a statement that expresses flaw/failure in
the reasoning process, while a reward criteria should be a statement that express
merits/goal accomplishment in the reasoning process.  Some of the final things that
your should CHECK BEFORE YOUR FINAL RESPONSE:
  - Try to make your rubrics specific to the question, but not so detailed as some of
the overly detailed metrics may not apply to all the response(e.g.  "fail to apply XXX
theorem" would be a acceptable metric, but "fail to reflect after miscalculating using
XXX theorem in step25" would be a very bad metric to avoid, since another students
response may not apply XXX theorem, nor having step 25).  In other words, do not let
the reference answer constrain your rubric system!  Again, DO NOT let the reference
answer constrain your rubric system!!!  To do this, try to propose a set of metrics
pretending the reference answer is not there, and reconsider adding some of them.
  - Remember, the answer above is only a reference answer from the student(not
necessarily a good response), so your rubric system should consider the merits already
presented in the response, and most importantly, room for further improvements.  To
enforce this, the reference response should have a score strictly below the average of
your proposed rubric system.  In order to check for this, you need to calculate the
maximum&minimum score from you rubric system, and score the given response at the end
of your json.  If you found that the rubrics you presented the first time gives a
overly-high score, brainstorm more rubrics pretending the reference answer is not there
in orderly to find room for improvements and add them to your rubric systems.  - Return
only the JSON object without additional commentary.
```

Figure 9: The prompt of the rubricator role (continued).

PROMPT:

```
You are an expert in evaluating student responses to math problems. Your task is to
assess a given response based on a set of binary rubrics and compute a final score.
[Task Input]:
- Question: The math problem the student is solving.
- Response: The student's step-by-step solution.
- Rubrics: A list of criteria for evaluation. Each criterion has:
  - category: The aspect of reasoning being assessed.
  - criterion: A binary statement (either a merit to reward or a flaw to penalize).
  - points: The points assigned (positive for rewards, negative for penalties).
[Task Instruction]:
Evaluate each criterion:
- For each criterion in the rubrics list (in the given order), determine if the
criterion is satisfied by the response.
  - Output True if the criterion is met (for a reward) or if the flaw is present (for a
penalty).
  - Output False otherwise.
- Base your judgement solely on the content of the response and the specific wording of
the criterion.
- Compute the final score: Sum the points of every criterion that is evaluated as True
(this includes both positive and negative points).
[Output Format]
Return a JSON object with the following structure:
"'json
{
  "judgement": [boolean, boolean, ...], // List of True/False for each criterion in
the same order as the rubrics
  "final_score": number // Sum of points for True judgements
}
"'
[Important Guidelines]
- Be objective: Do not consider any external knowledge beyond the provided question,
response, and rubrics.
- Binary decision: Each criterion must be evaluated as strictly True or False. There
are no partial credits.
- Order matters: The judgement list must exactly match the order of the rubrics
provided.
- Independent evaluation: Assess each criterion separately. The presence or absence of
one flaw/merit does not influence the evaluation of others (unless the criterion
explicitly references another).
  - Penalty criteria: For a criterion with negative points, True means the flaw is
present (so the negative points are added). False means the flaw is absent (so 0 is
added for that criterion).
  - Reward criteria: For a criterion with positive points, True means the merit is
present (so positive points are added). False means the merit is absent (so 0 is
added).
Now, proceed to evaluate the following:
Question: {question}
Response: {resp}
Rubrics: {rubrics}
Your final output should contain only the JSON object. Do not include any additional
text or explanations.
```

Figure 10: The prompt of the verifier.

E CASE STUDY

In this section, we present a case to illustrate how the rubricator proposes meaningful metrics conditioned on the reasoner's response. This example shows that the rubricator successfully identifies the key problem(*Employs ad-hoc manual listing for small digit sets instead of systematic permutation formula application, increasing error risk*) that caused miscalculation for $d = 0$ and $d = 1$ subsets, leading to the final error of 131. It also gives a hint to efficiently solving the problem and reducing reasoning redundancy (*Avoids redundant recounting by categorizing sets with minimum permutation $>$ 2013 upfront, reducing per-set analysis effort*). Therefore, the self-proposed rubrics can effectively guide the model to solve the specific problem, serving as valuable auxiliary reward for the reasoner.

Question: 2013 is the first year since the Middle Ages that consists of four consecutive digits. How many such years are there still to come after 2013 and before 10000?

Ground Truth: 149

Prediction: 131

Generated Rubrics:


```
{
    "question_domain":"math/digit-sequence-analysis",
    "rubrics":  [
        {
            "category":"Effectiveness of Problem Decomposition & Planning",
            "criterion":"Explicitly outlines digit set ranges, leading zero constraints, and
year boundary limits before permutation calculations",
            "points":3
        },
        {
            "category":"Reasoning Clarity & Flow",
            "criterion":"Uses hierarchical structuring (e.g., bullet points) for each digit
set with explicit permutation counts and exclusion justifications",
            "points":3
        },
        {
            "category":"Reasoning Focus & Efficiency",
            "criterion":"Avoids redundant recounting by categorizing sets with minimum
permutation > 2013 upfront, reducing per-set analysis effort",
            "points":3
        },
        {
            "category":"Effectiveness of BackTracking, Self-Validation & Error Handling",
            "criterion":"Fails to implement edge case validation (e.g., overcounting
leading-zero permutations) via secondary checklist reviews",
            "points":-4
        },
        {
            "category":"Other Question-Specific Aspects",
            "criterion":"Employs ad-hoc manual listing for small digit sets instead of
systematic permutation formula application, increasing error risk",
            "points":-2
        }
    ],
    "maximum_score":9,
    "minimum_score":-6
}
```


Figure 11: The rubricator's generated rubrics

Reasoner's Response:

Got it, let's try to figure out this problem: we need to find how many years after 2013 and before 10000 have four consecutive digits. So a four-digit year with four consecutive digits--wait, does that mean the digits are consecutive in order, like 2013 is 2,0,1,3? Wait no, 2013 isn't consecutive digits. The problem says "2013 is the first year since the Middle Ages that consists of four consecutive digits"--probably, it means the set of digits is four consecutive integers, maybe in any order?
First, what are four consecutive integers? Let's denote four consecutive digits as d, d+1, d+2, d+3 where d is from 0 to 6 (since d+3 $\leq$ 9). So possible d: 0–6.
d=0: {0,1,2,3}
d=1: {1,2,3,4}
d=2: {2,3,4,5}
d=3: {3,4,5,6}
d=4: {4,5,6,7}
d=5: {5,6,7,8}
d=6: {6,7,8,9}
A four-digit year can't start with 0, so we count permutations > 2013.
Set {0,1,2,3} (d=0): Total permutations 4! = 24, minus those starting with 0 (3! = 6), so 18. First digit 1: smallest is 1023 < 2013, ignore. First digit 2: smallest is 2013 (equal). Second digit 0: 2031 valid. Second digit 1: 2103, 2130 valid. Second digit 3: 2301, 2310 valid. So 5 numbers from this set.
Set {1,2,3,4} (d=1): All permutations valid (no 0). First digit 1: 1xxx < 2000, ignore. First digit 2: 2134, 2143, 2314, 2341, 2413, 2431, all > 2013. So 6 numbers.
==> Employs ad-hoc manual listing for small digit sets instead of systematic permutation formula application, increasing error risk
Set {2,3,4,5} (d=2): No 0, so all 4! = 24 permutations are valid four-digit. Smallest is 2345 > 2013, so all 24 valid.
Set {3,4,5,6} (d=3): All numbers $\geq$ 3456, all 24 valid.
Set {4,5,6,7} (d=4): all 24 valid.
Set {5,6,7,8} (d=5): all 24 valid.
Set {6,7,8,9} (d=6): all 24 valid.
==> Avoids redundant recounting by categorizing sets with minimum permutation > 2013 upfront, reducing per-set analysis effort
Now let's add them up:
Set {0,1,2,3}: 5
Set {1,2,3,4}: 6
Set {2,3,4,5}: 24
Set {3,4,5,6}: 24
Set {4,5,6,7}: 24
Set {5,6,7,8}: 24
Set {6,7,8,9}: 24
Calculations: 5 + 6 = 11; 11 + 24×5 = 11 + 120 = **131**.

Figure 12: The reasoner's response

F BROADER IMPACTS

Potential benefits. Improving chain-of-thought reasoning through rubric-based RL could support education (*e.g.,* tutoring systems that reward structured thinking), scientific and engineering assistance (*e.g.,* math and coding workflows), and debugging of model reasoning when failures are tied to interpretable rubric criteria rather than only final answers.

Potential risks. Stronger reasoning models may lower the cost of misuse (*e.g.,* automated cheating on graded assessments, crafting persuasive misinformation with plausible step-by-step narratives, or aiding prohibited tasks if deployed without safeguards). Models trained with learned rubrics may still encode biases present in prompts, rollouts, or verifier behavior.

Mitigations. Risk-aware deployment should combine usage policies, monitoring for policy violations, human oversight for high-stakes decisions, and verification layers independent of the trained policy; releasing artifacts should follow organizational review and applicable license terms for underlying models and data.

G ASSET LICENSES

Table 7 summarizes publicly stated licenses for major third-party assets referenced in this paper (see Hugging Face model/dataset cards or repository `LICENSE` files for authoritative wording). Contest-derived evaluation protocols (AIME/AMC-style tasks) should be used under the rules and citations indicated by the respective benchmark papers and organizers.

Table 7: Third-party asset licenses (informative summary; verify against live artifacts).

Asset	**License / terms (as distributed)**
Qwen3 base models (e.g., Qwen3-4B/8B-Base on Hugging Face)	Apache License 2.0 (model card / repository).
DAPO-Math-17k (Yu et al., 2025a)	Apache License 2.0 on multiple community mirrors (verify against the exact HF snapshot used).
GPQA / GPQA-Diamond (Rein et al., 2023)	Dataset on Hugging Face under CC BY 4.0 with usage notes; companion GitHub code often under MIT—follow dataset card.
SuperGPQA (M-A-P Team, 2025)	Open Data Commons Attribution License (ODC-BY) on Hugging Face (`m-a-p/SuperGPQA`).
LlamaFactory (GitHub `hiyouga/LlamaFactory`)	Apache License 2.0 (repository `LICENSE`).
verl	Apache License 2.0 (GitHub `volcengine/verl`).