Stepwise Think-Critique: Interleaved Reasoning and Self-Critique in a Single LLM

Anonymous ACL submission

Abstract

Human beings solve complex problems through critical thinking, where reasoning and evaluation are intertwined to converge toward correct solutions. However, most existing large language models (LLMs) treat the reasoning and verification as separate processes: they either generate reasoning without explicit self-checking or rely on external verifiers to detect errors post hoc. The former lacks immediate feedback, while the latter increases system complexity and hinders synchronized learning. Motivated by human critical thinking, we propose **Stepwise Think-Critique (STC)**, an end-to-end trainable framework in which a single LLM emits a structured, step-level critique inline with each reasoning step. STC is trained with reinforcement learning that complements reasoning rewards with a critique-consistency reward derived from final-answer correctness, jointly optimizing reasoning correctness and critique reliability. On five mathematical reasoning benchmarks, STC improves Pass@1 by **7.2%** over the 1.5B base model, attains **67.4%** step-level critique F1, and supports critique-guided test-time scaling that beats Majority Voting by up to **32%** on AIME24—a step toward **LLMs with built-in critical thinking**.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in solving complex reasoning tasks. Techniques such as Chain of Thought (CoT) prompting (Wei et al., 2022) and reinforcement-based reasoning approaches (*e.g.*, DeepSeek-R1) (Guo et al., 2025) enable models to decompose problems into multi-step reasoning trajectories, leading to improved performance in domains such as mathematical problem-solving, program synthesis, and complex question-answering (Guo et al., 2025; Yu et al., 2025; Chen et al., 2025).

Despite these advances, a fundamental limitation persists: current approaches lack an intrinsic, on-the-fly mechanism to assess the correctness of intermediate reasoning steps. Such a mechanism is highly desirable because i) in domains such as mathematics, science, and finance, users require transparent judgments on whether each reasoning step is correct; ii) integrated self-critique allows models to adapt their reasoning trajectories dynamically, enabling self-assessment and effective test-time scaling during inference; and iii) stepwise evaluation could provide dense feedback, facilitating more effective policy optimization.

Most existing reasoning models (Guo et al., 2025; OpenAI, 2025) focus solely on generating reasoning and final answers, as illustrated in Figure 1 (a). Some studies have explored post-hoc verification, where an external verifier evaluates the correctness of the reasoning process (see Figure 1 (b)). For example, process reward models (PRMs) have been developed to evaluate intermediate reasoning steps produced by a policy model (Zhang et al., 2025; Zou et al., 2025; Khalifa et al., 2025; Xiong et al., 2025). Although such methods can detect errors, the post-hoc nature (*i.e.*, the separation of reasoning and verification) prevents timely feedback, and maintaining separate reasoning policy and verifier models increases system complexity, hindering seamless deployment in real-world pipelines.[1]

The ability to interleave reasoning and evaluation is a hallmark of human problem solving. Humans naturally engage in critical thinking, continuously reflecting on intermediate conclusions, assessing their validity, and adjusting subsequent steps accordingly. This process allows humans not only to reach correct solutions more efficiently but also to produce reasoning traces that are interpretable and trustworthy. Inspired by this, we propose a novel framework called Stepwise Think-Critique (STC). As illustrated in Figure 1 (c), un-

[1] In deployment, passing trajectories between models requires additional scheduling, memory management, and fault tolerance burdens, which increases system fragility.

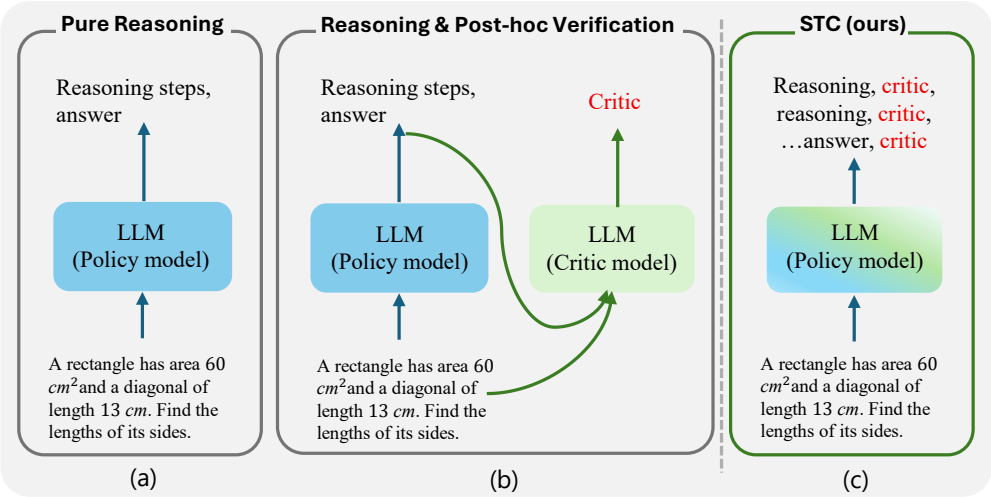


Figure 1: **Paradigms of reasoning and verification in LLMs.** (a) *Pure reasoning*: a model generates multi-step reasoning and answers without assessment. (b) *Post-hoc verification*: reasoning is produced first and then critiqued by an external verifier (*e.g.*, process reward model), which provides delayed feedback and increases system complexity. (c) *Stepwise Think-Critique* (***ours***): a single model outputs interleaved reasoning and critique at each step, enabling timely self-assessment, effective test-time scaling, and improved interpretability.

like existing paradigms that separate reasoning and verification, our method unifies them. At each step of problem solving, the model not only generates the next reasoning step but also the justification for its correctness. We instantiate our framework within an LLM that interleaves reasoning and self-critique steps. The model is trained with RL using a hybrid objective that combines reasoning rewards (based on the final outcomes), critique-consistency rewards (encouraging the model's self-assessment over the final answer to align with the ground-truth correctness), and format rewards for structured outputs. STC endows the model with reasoning and critiquing capabilities.

Our work makes the following contributions:

- We propose **Stepwise Think-Critique (STC)**, a framework in which a single LLM emits a structured, step-level critique *inline with* each reasoning step, eliminating the post-hoc reasoner–verifier split.
- We design an **end-to-end RL training scheme** that jointly optimizes reasoning and critique within a single model: a critique-consistency reward derived from final-answer correctness is propagated only to critique tokens via masking, for accurate credit assignment.
- On five math benchmarks, STC improves Pass@1 by **7.2%**, attains **67.4%** step-level critique F1 on average, and beats Majority Voting by up to **32%** on AIME24.

Our unified framework eliminates the requirement of an external verifier and enables on-the-fly self-critique and effective test-time scaling. This represents a crucial step towards **building LLMs with built-in critical thinking**, bridging the gap between human-like problem solving and reliable machine reasoning. At inference time, we can optionally enable the critiquing for enhanced interpretability—providing judgments on where and why mistakes occur—or disable it to reduce token usage.

2 Related Work

2.1 RL for LLM Reasoning

Reinforcement learning (RL) has become a central paradigm for eliciting multi-step reasoning—"thinking"—in LLMs, with works such as DeepSeek-R1 (Guo et al., 2025), DAPO (Yu et al., 2025), and REINFORCE++ (Hu, 2025) reinforcing sequences of reasoning actions that lead to correct solutions.

Actor–critic methods like VAPO (Yue et al., 2025) use a value model to estimate expected return but do not explicitly assess the correctness of reasoning steps or provide interpretable feedback. Existing RL frameworks thus lack intrinsic, fine-grained self-assessment capabilities.

A separate line of work studies LLM *self-correction*, where an explicit refinement pass consumes a response-level critique signal—Self-Refine (Madaan et al., 2023), Self-Verification (Weng et al., 2023), CRITIC (Gou

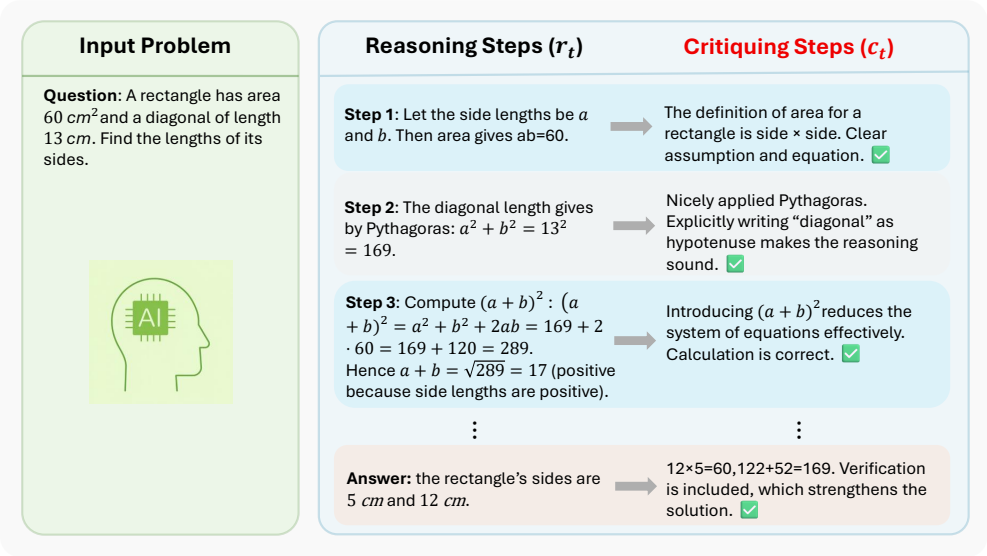


Figure 2: Illustration of **Stepwise Think-Critique (STC).** Given an input problem, the model alternates between generating a reasoning step (r_t) and a critiquing step (c_t). Critiques provide immediate correctness assessment, which supports on-the-fly error localization. At inference time, users may choose *compact mode* (reasoning only) or *full mode* (reasoning+critique) depending on efficiency vs. interpretability requirements.

et al., 2024) (tool-augmented), and SCoRe (Kumar et al., 2024). DeepSeek-R1 (Guo et al., 2025) shows reflection-style "wait" tokens can emerge mid-trajectory from outcome-only RL, while such reflection is unstructured and unsupervised.

STC pursues a different objective: equipping a single model with an intrinsic, consistency-supervised, step-level critique capability, directly enabling error localization, best-of-K selection, and interpretability without an external verifier.

2.2 Process Reward Modelling

Process reward models (PRMs) provide another perspective on stepwise evaluation, focusing on the intermediate reasoning steps rather than solely on the final answer (Zhong et al., 2025). By offering finer-grained feedback, PRMs could improve credit assignment, help identify local errors.

Designs of PRMs have evolved along multiple axes. Early approaches adopt a discriminative format, where a classification head predicts step-level correctness scores (Lightman et al., 2023; Zhang et al., 2025). More recent works embrace generative approaches, producing natural language judgments for each reasoning step (Zhang et al., 2024; Zheng et al., 2025; Khalifa et al., 2025; Zhao et al., 2025). ThinkPRM (Khalifa et al., 2025), for instance, generates a verification chain-of-thought (CoT) that critiques and justifies each step post hoc, improving interpretability and error localization.

PRMs are typically applied in two main ways. First, at inference, they can rank multiple solution candidates, selecting the most reliable one (Lightman et al., 2023). Second, during policy optimization, PRMs provide step-level reward signals that guide model updates. Despite their utility, most PRMs are trained offline on curated datasets and remain frozen, which can make them brittle under distribution shifts. PRIME (Cui et al., 2025) mitigates this limitation by deriving implicit process rewards from outcome labels through SFT, while TANGO (Zha et al., 2025) jointly trains a generator and PRM via RL. Nevertheless, these methods rely on separate models, which adds complexity, communication overhead, and memory usage. *In contrast, STC integrates the critic into a single model, avoiding extra communication and deployment overhead.*

3 Stepwise Think-Critique (STC)

3.1 Overview of STC

We introduce **Stepwise Think-Critique (STC)**, a unified framework that interleaves reasoning and critique within an output sequence. Unlike prior paradigms that either perform pure reasoning or rely on external post-hoc verification, STC equips a single model with the ability to generate reasoning steps and simultaneously assess their correctness.

Formally, given an input problem x, the model generates a sequence that alternates between reasoning step $\{r_t\}_{t=1}^{T}$ and critique step $\{c_t\}_{t=1}^{T}$,

where T denotes the number of steps that varies for different sequences. As illustrated in Figure 2, each reasoning step r_t advances the solution, while the corresponding critique c_t provides a natural-language judgment and score for assessing the correctness of r_t. The final solution is produced in the last reasoning step, followed by its associated critique.

We first train the model with supervised fine-tuning (SFT) on our synthesized reasoning–critique data, enabling it to learn the interleaved reasoning-critique patterns and to acquire basic critique ability (for cold start). We then further train the model with RL using reasoning reward, critique-consistency reward, and format reward, to boost critical thinking capability.

3.2 Interleaved Reasoning-Critiquing

For an LLM, to enable interleaved generation, we design instruction formats that explicitly encourage the model to alternate between reasoning and critique. Specifically, the prompt specifies the required structure:

$$r_1 \to c_1 \to r_2 \to c_2 \to \cdots \to r_T \to c_T. \quad (1)$$

Reasoning steps are partitioned according to the generated `\n\n`. Each critique step is expressed in natural language and consists of two parts: (i) a concise justification, and (ii) a binary correctness label, where 1 denotes "Correct" and 0 denotes "Incorrect." The model outputs critiques using the following tagged format: `<critic>`{*justification*}`</critic>` and a binary judgment in the format `<score>`{*1 or 0*}`</score>`. This structured format allows the model to explain errors in context while keeping the output compact and informative.

During inference, through prompting alone, we support two modes:

- **Compact mode**: reasoning only, omitting critiques to save tokens.
- **Full mode**: reasoning and critique are interleaved, providing detailed step-level judgment, enabling error localization and enhancing interpretability.

This dual-mode design offers a flexible trade-off between efficiency and interpretability, allowing users to select the mode that best suits their needs.

3.3 Supervised Fine-Tuning (SFT)

We begin with supervised fine-tuning (SFT) to equip the model with interleaved reasoning-critique generation. We synthesize SFT data by sampling reasoning trajectories $(r_1, \ldots, r_T)$ from the base model on training problems, then prompting a strong LLM (*e.g.*, GPT-5) to produce stepwise critiques $c_n = (j_n, s_n)$ following the format in §3.2, where j_n is a natural-language justification and $s_n \in \{0, 1\}$ a binary correctness label. To remove unreliable supervision, a rule-based verifier V checks the final answer r_T, and we discard trajectories where the strong LLM's score on r_T disagrees with V. We then fine-tune the model on the resulting $\mathcal{D} = \{(x, \tau)\}$ with $\tau = (r_1, c_1, \ldots, r_T, c_T)$ by minimizing $\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,\tau)\sim\mathcal{D}} \log \pi_\theta(\tau \mid x)$.

3.4 RL with Joint Reasoning and Critiquing

While SFT provides initial alignment, it is limited by data coverage and generalization. We therefore further post-train the model using Grouped Reinforcement Policy Optimization (GRPO) with designed rewards.

3.4.1 Reward Design

We define rewards at three levels:

***Reasoning reward**:* We assign reward based on whether the answer is correct or not:

$$R_{\text{reason}}(\tau) = \mathbb{1}[r_T = y], \quad (2)$$

where y is the ground-truth answer. $\mathbb{1}[\cdot]$ denotes indicator function.

***Critique-consistency reward (Critique reward)**:* Since no step-level ground-truth critiques are available during training, we derive supervision from the correctness of the final answer. Let $z \in \{0, 1\}$ be the true correctness indicator of the final answer r_T, *i.e.*,

$$z = \mathbb{1}[r_T = y]. \quad (3)$$

Recall that s_T denotes the binary label predicted by the critique for the final answer (*i.e.*, reasoning step r_T). We then define the critique reward w.r.t. the final answer as

$$R_{\text{crit}}(c_T) = \mathbb{1}[s_T = z], \quad (4)$$

which *encourages the model to produce critiques consistent with the actual correctness of the final answer. Similar to the RL mechanism that the final reasoning reward can drive the learning of intermediate reasoning steps (Guo et al., 2025), our critique reward drives the learning of intermediate critic steps, which naturally penalizes false judgments that do not lead to correct final critique.*

Format reward: To ensure structural validity, we require that each reasoning step is followed by a well-formed critique (*i.e.*, both justification and binary label). Let $v_n \in \{0, 1\}$ be an indicator of whether the n-th reasoning–critique pair is correctly formatted. The format reward is defined as the fraction of valid pairs among all T reasoning steps:

$$R_{\text{format}}(\tau) = \frac{1}{T} \sum_{n=1}^{T} v_n, \tag{5}$$

where $R_{\text{format}}(\tau) \in [0, 1]$ encourages the model to consistently produce structurally correct interleaved outputs.

3.4.2 Optimization

We leverage GRPO to update model π_θ to maximize expected advantages. For an input problem x, the policy model samples a group of G individual trajectories $\{\tau_k\}_{k=1}^{G}$. *To ensure correct credit assignment, we perform separate optimization for the critique parts.*

Specifically, the critique advantage for the k-th trajectory is computed by normalizing the group-level critique-consistency rewards:

$$A_{\text{crit}}^{(k)} = \frac{R_{\text{crit}}^{(k)} - \text{mean}\left(\{R_{\text{crit}}^{(k)}\}_{k=1}^{G}\right)}{\text{std}\left(\{R_{\text{crit}}^{(k)}\}_{k=1}^{G}\right)}. \tag{6}$$

The gradients from critique-consistency rewards are only back-propagated to **critique tokens** within $\{c_n^{(k)} | n = 1, 2, \cdots, T\}$ for accurate credit assignment. Therefore, the critique-consistency advantage for the t-th token in τ_k is

$$\mathcal{A}_{\text{crit}}^{k,t} = A_{\text{crit}}^{(k)} \cdot \mathbb{1}[t \in \text{Critique}(\tau_k)], \tag{7}$$

where $t \in \text{Critique}(\tau_k)$ denotes that the t-th token belongs to critique tokens.

Considering the critiques would influence the next step reasoning, the reasoning reward and format reward are used to optimize all the tokens. Similar to (6), we obtain reasoning advantage $A_{\text{reason}}^{(k)}$ and format advantage $A_{\text{format}}^{(k)}$, respectively.

The combined advantage is then:

$$\mathcal{A}_{k,t} = \lambda_{\text{crit}} \alpha \mathcal{A}_{\text{crit}}^{k,t} + \lambda_{\text{reason}} A_{\text{reason}}^{(k)} + \lambda_{\text{format}} A_{\text{format}}^{(k)}, \tag{8}$$

where λ_{crit}, λ_{reason}, λ_{format} control contributions; $\alpha = \sum_k |\tau_k| / \sum_k |\tau_k^{\text{crit}}|$ undoes the dilution from GRPO's token-level averaging (Eq. 9 in Appendix A), since $\mathcal{A}_{\text{crit}}^{k,t}$ is supported on only $|\tau_k^{\text{crit}}|$ tokens.

We optimize $\mathcal{A}_{k,t}$ via DAPO-style token-level GRPO (Yu et al., 2025) with a KL constraint to a reference policy (Appendix A, Eq. 9).

This joint objective encourages accurate reasoning, faithful critiques, and well-structured outputs, enabling the model not only to solve problems but also to verify and explain its reasoning process.

4 Experiments

4.1 Experimental Setup

4.1.1 SFT Data Construction

We construct the SFT dataset by sampling 10k base-model rollouts on the training pool, generating stepwise critiques with GPT-5, and filtering for structural integrity and answer-critique consistency, yielding **5,056** interleaved samples ($\sim$51% retention). Length statistics and the full filtering pipeline are detailed in Appendix A.

4.1.2 Implementation Details

We use DS-Qwen-1.5B (Guo et al., 2025) as the 1.5B-parameter base model and train on DeepScaleR-Preview-Dataset (Luo et al., 2025). Hyperparameters, max response lengths, and the system prompt are in Appendix A.

4.1.3 Evaluation Protocol

We evaluate both the reasoning and critique capability of models. Evaluations are performed with a maximum response length of 8,192 tokens for reasoning and 4,096 tokens for critique by default.

For reasoning quality, we report **Pass@1** (P@1 for short) and **Pass@8**.

For critique quality, we evaluate both the **answer level** and the **process level**. At the answer level, correctness is directly **verifiable** from the final solution (see Eq. 4). At the process level, because step-level human labels are unavailable, we use a strong LLM judge (GPT-5-mini) to produce pseudo labels for stepwise critiques.

We report three metrics: **Correct Accuracy (Corr.)**, **Error Accuracy (Err.)**, and their harmonic mean **F1**; higher is better for all (↑). Corr. measures how well the critic accepts truly correct items, Err. measures how well it rejects truly incorrect items. Following (Zheng et al., 2025), we **use F1 to assess the overall critic capability**. This metric penalizes degenerate solutions that achieve high correct accuracy but fail to detect errors, or vice versa. Detailed metric definitions (including step-level and answer-level) are provided in Appendix B.

Model	AIME24		AMC23		MATH-500		Minerva		OlympiadBench		Average	
	P@1	P@8	P@1	P@8	P@1	P@8	P@1	P@8	P@1	P@8	P@1	P@8
DS-Qwen-1.5B	17.5	43.3	51.1	72.3	76.2	89.6	25.1	46.3	36.0	52.2	41.2	60.7
STC-SFT	18.3	46.7	49.1	71.1	73.3	88.6	20.3	39.7	34.2	49.3	39.1	59.1
STC (compact)	**27.9**	**63.3**	**62.7**	**81.9**	82.3	**93.4**	27.8	48.5	41.8	57.2	**48.5**	**68.9**
STC (full)	**27.9**	56.7	60.5	**81.9**	**82.9**	92.6	**27.9**	**49.3**	**42.7**	**57.3**	48.4	67.6

Table 1: Reasoning performance (%) of models on math reasoning benchmarks, evaluated with Pass@1 (P@1) and Pass@8 (P@8). **Bold** marks the best per column.

Model	AIME24			AMC23			MATH-500			Minerva			OlympiadBench			Average		
	Corr.	Err.	**F1**	Corr.	Err.	**F1**	Corr.	Err.	**F1**	Corr.	Err.	**F1**	Corr.	Err.	**F1**	Corr.	Err.	**F1**
STC-SFT	100	10.5	19.0	98.5	22.6	36.8	99.2	7.4	13.8	96.6	8.7	16.0	98.5	7.8	14.4	98.8	11.4	20.0
STC w/o Critic	97.1	25.7	40.7	95.2	26.7	41.7	97.4	16.6	28.4	94.6	12.7	22.4	94.2	24.2	38.5	95.7	21.2	34.3
STC	82.1	60.3	**69.6**	90.3	55.1	**68.4**	96.2	32.6	**48.7**	90.1	21.9	**35.2**	90.5	43.1	**58.4**	89.8	42.6	**56.1**

Table 2: **Answer-level critic** performance (%) evaluated against **verifiable ground truth**. Corr. denotes correct-answer accuracy (↑). Err. denotes error-answer accuracy (↑). F1 (↑), as the overall evaluation, is the harmonic mean of Corr. and Err.

4.1.4 Evaluation Datasets

We evaluate our framework on multiple mathematical reasoning benchmarks: **AIME24**, **MATH-500**, **AMC23**, **Minerva**, and **OlympiadBench**. These datasets vary in problem complexity and problem styles, providing a comprehensive testbed.

4.2 Results

We evaluate STC on three axes: intrinsic reasoning and critique quality (§4.2.1, §4.2.2). We compare three model variants: *DS-Qwen-1.5B* (base model), *STC-SFT* (after supervised fine-tuning), and *STC* (RL-trained). We then study downstream use of the self-critic for inference-time selection (§4.2.3), and benchmark critique quality against seven PRMs (§4.2.4). §4.2.5 shows qualitative trajectories.

4.2.1 Reasoning Performance

Table 1 reports reasoning accuracy across all benchmarks. **i)** SFT reduces average Pass@1 accuracy by 2.1% relative to the base model. We attribute this drop to enabling stepwise critique, which shifts the model's output distribution and temporarily degrades reasoning performance. **ii)** Subsequent RL training recovers and surpasses this gap: *STC (full)* outperforms *STC-SFT* by **9.3%** at Pass@1 and **8.5%** at Pass@8 on average, and also exceeds the base DS-Qwen-1.5B by **7.2%** Pass@1, showing that the inline-critic objective is compatible with strong reasoning.

Full mode vs. compact mode. STC can operate in two inference modes: *compact mode*, where critique generation is disabled and the model outputs only reasoning, and *full mode*, where the model performs both reasoning and stepwise critique. The two modes share the same model weights and differ *only* in the system prompt (highlighted in orange in Tables 11 and 15). As shown in Table 1, both modes outperform STC-SFT significantly and achieve comparable reasoning accuracy: since the reasoning capability is acquired during training, toggling critique generation at inference time does not affect it. Users can flexibly choose the mode based on whether intermediate critiques are needed, without sacrificing reasoning performance.

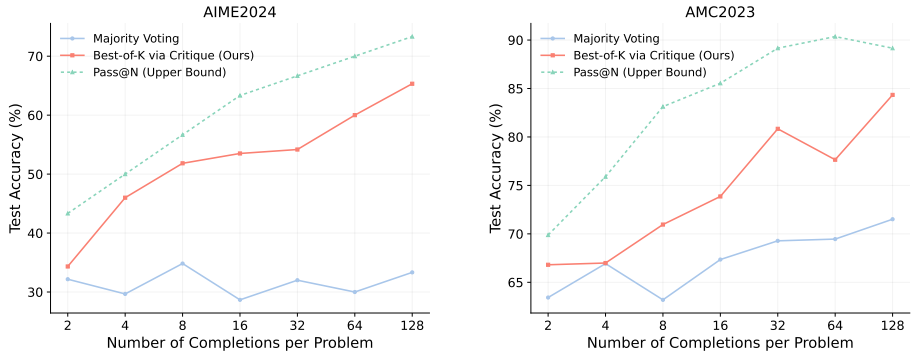
4.2.2 Critique Performance

We evaluate critique quality using Corr. (accuracy on truly correct instances), Err. (accuracy on truly incorrect instances), and their harmonic mean F1 at both answer-level (Table 2) and step-level (Table 3). We compare our final model *STC* with two variants: *STC-SFT* (SFT only), and *STC w/o Critic* (RL without critique reward).

Answer-level critique (Table 2). **i)** *STC-SFT* exhibits a strong positive bias: it achieves 98.8% Corr. but only 11.4% Err., indicating poor error detection. Table 6 in Appendix C.2 shows that only 4% of answers are predicted as incorrect, while 27% are actually incorrect from the valid samples. **ii)** Adding RL with only answer and format rewards (*i.e.*, *STC w/o Critic*) modestly increases Err. to 21.2%, but overall F1 remains low. **iii)** Incorporating the critique-consistency reward (*i.e.*, *STC*) substantially improves error detection, **raising Err. from 21.2% to 42.6%**. Although Corr. drops

Model	AIME24			AMC23			MATH-500			Minerva			OlympiadBench			Average		
	Corr.	Err.	F1	Corr.	Err.	F1	Corr.	Err.	F1	Corr.	Err.	F1	Corr.	Err.	F1	Corr.	Err.	F1
STC-SFT	93.6	35.4	51.4	94.5	31.3	47.1	97.3	24.2	38.8	95.4	19.3	32.2	90.6	45.0	60.1	94.3	31.1	45.9
STC w/o Critic	94.0	35.2	51.2	94.8	30.0	45.6	96.1	25.0	39.7	94.6	20.4	33.5	89.9	47.3	61.9	93.9	31.6	46.4
STC	84.4	61.3	**71.0**	84.0	59.5	**69.7**	89.7	53.3	**66.9**	89.6	37.3	**52.7**	77.9	75.3	**76.6**	85.1	57.3	**67.4**

Table 3: **Process-level critic** performance (%) evaluated by GPT-5-mini. To save cost, we tested on the first 30 prompts $\times$ 8 rollouts $=$ 240 trajectories/samples for each dataset.



(a) (b)

Figure 3: Test-time scaling on **(a)** AIME24 and **(b)** AMC23, comparing Pass@N (oracle upper bound), Majority Voting, and our Best-of-K via Critique. Our critique-based selection scales more effectively with K and consistently outperforms Majority Voting.

about 6%, **the overall F1 improves markedly from 34.3% to 56.1%**, indicating a much better balance between recognizing correct and incorrect answers. Table 6 in Appendix C.2 shows that the predicted negative ratio increases from 4% (*STC-SFT*) to 21% (*STC w/ Critic*), approaching the ground-truth ratio of 29%.

Step-level critique (Table 3). A similar trend is observed for the step-level critic. **i)** *STC-SFT* shows a strong positive bias (94.3% Corr., 31.1% Err.). Table 7 in Appendix C.2 shows that only 11% of steps are predicted as erroneous, despite a ground-truth error rate of about 20%. *STC w/o Critic* behaves similarly. **ii)** With the critique reward, *STC* **substantially improves error detection, raising Err. from 31.6% to 57.3%** and **F1 from 46.4% to 67.4%**, revealing that our hybrid rewards effectively encourage balanced and reliable step-level critique. Table 7 in Appendix C.2 shows that the predicted error-step ratio increases from 11% to 23%, approaching the ground-truth ratio of 19%.

Reliability of the GPT-5-mini judge. Since human-annotated step-level labels are unavailable, we employ GPT-5-mini as an automatic judge for process-level evaluation, which may introduce noise. To assess its reliability, we compare GPT-5-mini judgments with verifiable answer-level ground truth on a subset, obtaining an agreement rate of about **90%**. This suggests that GPT-5-mini provides reasonably reliable pseudo ground truth for stepwise critique evaluation.

To further validate the reliableness of GPT-5-mini step labels, we use a stronger (more expensive) LLM, Claude Opus 4.7 (with maximum reasoning budget and no truncation), as a reference and re-evaluate the AIME24 labeled corpus. Treating Claude as the gold standard, GPT-5-mini achieves a balanced **F1 of 87.3%** (Corr. 88.3%, Err. 86.4%, raw agreement 88.1%). To balance performance and cost, we use GPT-5-mini as the judge. More analysis can be found in Appendix C.1.

4.2.3 Critique-Guided Test-Time Scaling

We evaluate critique-guided test-time scaling by independently testing different K. Our method (Best-of-K via Critique) selects solutions whose final answer critique score is 1, and if multiple candidates remain, applies Majority Voting. We compare against standard Majority Voting and report Pass@N as a practical upper bound (oracle selection).

As shown in Figure 3, our critique-based selection consistently outperforms Majority Voting at the same sampling budget, with gains of 2.2%–32.0% on AIME24 and 0%–12.8% on AMC23. **These results show that our model has learned**

Method	AIME24	AMC23	MATH-500	Minerva	OlympiadBench	Average
Math-Shepherd-7B (Wang et al., 2024)	66.3	64.8	70.8	61.8	61.4	65.0
RLHFlow-PRM-8B (Dong et al., 2024)	58.6	58.4	66.5	58.7	58.9	60.2
Skywork-PRM-1.5B (Skywork o1 Team, 2024)	51.0	53.4	68.3	53.7	49.5	55.2
Skywork-PRM-7B (Skywork o1 Team, 2024)	47.7	52.8	68.1	54.3	46.8	53.9
Qwen-Math-PRM-7B (Zhang et al., 2025)	62.4	66.0	**74.7**	**64.5**	61.9	65.9
PathFinder-PRM-7B (Pala et al., 2025)	61.5	60.9	69.1	57.3	56.7	61.1
Granite-PRM-8B (IBM Research, 2026)	59.7	59.9	66.6	59.9	56.9	60.6
STC (1.5B, calib-free)	**71.0**	**69.7**	66.9	52.7	**76.6**	**67.4**

Table 4: Step-level critique F1 (%) comparison of STC and external PRMs. Each external PRM cell reports F1 at its per-dataset **oracle** threshold; STC is calibration-free with a single binary prediction per step.

effective critique capability, providing reliable critic signals for solution selection. Note that as K increases, Majority Voting is not guaranteed to improve monotonically, because additional samples can amplify a dominant but incorrect answer pattern rather than correct it.

We also explored stepwise critiques as a dense RL reward; the formulation we tried did not yield robust gains (see Appendix C.5).

4.2.4 Comparison with External Process Reward Models

We compare STC against seven PRMs: Math-Shepherd-Mistral-7B (Wang et al., 2024), RLHFlow-PRM-Deepseek-8B (Dong et al., 2024), Skywork-o1-PRM-1.5B/7B (Skywork o1 Team, 2024), Qwen2.5-Math-PRM-7B (Zhang et al., 2025), PathFinder-PRM-7B (Pala et al., 2025), and Granite-3.3-8B Math PRM v2 (IBM Research, 2026).

All systems are evaluated on the same 1,200 STC reasoning trajectories (by dropping critics) (30 prompts $\times$ 8 rollouts $\times$ 5 datasets) against the same GPT-5-mini pseudo-GT step labels. Each PRM emits a continuous score per step; we sweep $\sim$200 binarization thresholds in $[0.001, 0.999]$ and report the per-dataset **oracle** threshold (F1-maximizing – an upper bound requiring labeled tuning per dataset). STC is calibration-free, emitting a single binary score per step.

Table 4 shows the step-level performance comparison. STC reaches **67.4%** average step-F1, ahead of the strongest PRM at its oracle threshold (Qwen-Math-PRM-7B, 65.9%), and is best on 3 of 5 benchmarks (AIME24, AMC23, OlympiadBench) – despite being a 1.5B model with no per-dataset threshold tuning. Figure 4 in Appendix C.3 shows the full Corr. vs. Err. curves, where STC achieves the best performance on average. Beyond accuracy, STC is also more efficient: it saves $\sim$28%/$\sim$75–77% FLOPs and $\sim$49%/$\sim$82–84% peak GPU memory versus the 1.5B/7B–8B PRM pipelines (Appendix C.3, Table 8).

Note on ProcessBench compatibility. We attempted ProcessBench (Zheng et al., 2025) via post-hoc verification prompts, but the 1.5B base model is too small to reliably emit ProcessBench's required error-index format; ProcessBench compatibility likely requires scaling to a larger backbone, which we leave for future work.

4.2.5 Qualitative Analysis

Full mode: Some typical full-mode trajectories from STC are shown in Tables 11 to 14 in Appendix C.6. Critiques are informative yet concise: they provide explicit error causes for incorrect steps and brief confirmations for correct steps. Table 11 shows STC catching a wrong intermediate guess $(1/\binom{6}{2})$. Table 13 shows STC identifying the cause of an error (a misread problem constraint).
Compact mode: The same problems in compact mode (reasoning only) are shown in Tables 15 to 18 in Appendix C.7.
Failures: STC's critic is not always reliable. Table 19 in Appendix C.8 shows a trajectory where STC missed a wrong unit conversion ($3\,\mathrm{Mpc} \rightarrow 3000\,\mathrm{pc}$) at step 3.

5 Conclusion

We presented Stepwise Think-Critique (STC), a unified framework that interleaves reasoning and self-critique within a single model. Across multiple mathematical reasoning benchmarks, STC demonstrates strong reasoning and critiquing capabilities and produces more interpretable process traces. These results support STC as a practical step toward LLMs with built-in critical thinking, bridging the gap between human-like problem solving and reliable machine reasoning.

We hope this work motivates further research on critical-thinking mechanisms that jointly improve transparency, reliability, and reasoning quality.

6 Limitations

While STC demonstrates promising results on mathematical reasoning, several limitations remain.

Scale and single-seed evaluation. We validate the framework on a single 1.5B-parameter base model (DS-Qwen-1.5B) under a fixed random seed; multi-seed experiments were not run due to compute cost (one full SFT+RL run requires $\sim$5 days on 16$\times$40GB A100). Extension to 7B/14B+ models and to multimodal settings is left as future work.

Domain coverage. Our experiments cover only text-only English mathematical reasoning across five benchmarks. We do not evaluate STC on code synthesis, agentic / tool-use tasks, multilingual reasoning, multimodal (vision+math) problems, or open-ended generation. While the framework itself is domain-agnostic, our claims about "critical thinking" should be read narrowly as referring to step-level error detection in mathematical chain-of-thought.

Critic reliability. STC's step-level balanced F1 of 67.4% leaves substantial room for improvement, especially on Minerva (52.7% F1) where critic reliability is weakest.

Step granularity vs. PRMs. STC's `\n\n`-delimited atomic steps are substantially finer than the paragraph-level granularity of most external PRMs. The PRM comparison (Table 4) therefore operates at different base rates across systems and should be read as evidence that STC reaches a strong calibration-free operating point, not as a direct step-by-step claim over PRMs in their native granularity.

7 Ethical Considerations

STC is a methodology study evaluated on standard public mathematical reasoning benchmarks; it does not introduce new user-facing systems, sensitive datasets, or task domains beyond those already established in the LLM reasoning literature. Nevertheless, we identify a concrete risk that warrants disclosure.

Critique reliability. As shown in Table 2 of the main manuscript, STC's error-detection accuracy on the answer level remains well below 100% (averaging 42.6%); a positive critique should therefore not be interpreted as a correctness guarantee, particularly in high-stakes settings such as education, tutoring, or scientific computing where users may over-trust an explicit score-1 signal.

We do not foresee misuse pathways unique to STC beyond those already common to LLM-based mathematical assistants.

References

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, and 1 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. RLHFlow: Open recipes for RLHF reward modeling. https://github.com/RLHFlow/RLHF-Reward-Modeling. RLHFlow/Llama3.1-8B-PRM-Deepseek-Data.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.

IBM Research. 2026. Granite 3.3 math PRM v2: An 8b process reward model with 128k context. https://huggingface.co/ibm-granite/granite-3.3-8b-math-prm-v2.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, and 1 others. 2024. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *International Conference on Learning Representations*.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

OpenAI. 2025. Introducing openai o3 and o4-mini. OpenAI.

Tej Deep Pala, Panshul Sharma, Amir Zadeh, Chuan Li, and Soujanya Poria. 2025. Error typing for smarter rewards: Improving process reward models with error-aware hierarchical supervision. *Preprint*, arXiv:2505.19706. PathFinder-PRM-7B.

Skywork o1 Team. 2024. Skywork-o1 open series. https://huggingface.co/Skywork/Skywork-o1-Open-PRM-Qwen-2.5-7B. Skywork-o1-Open-PRM-Qwen-2.5-1.5B and 7B.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP*.

Wei Xiong, Wenting Zhao, Weizhe Yuan, Olga Golovneva, Tong Zhang, Jason Weston, and Sainbayar Sukhbaatar. 2025. Stepwiser: Stepwise generative judges for wiser reasoning. *arXiv preprint arXiv:2508.19229*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. DAPO: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S Boning, and Dina Katabi. 2025. Rl tango: Reinforcing generator and verifier together for language reasoning. *arXiv preprint arXiv:2505.15034*.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and 1 others. 2025. Genprm: Scaling test-time compute of process reward models via generative reasoning. *arXiv preprint arXiv:2504.00891*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of Association for Computational Linguistics*, pages 1009–1024.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *arXiv preprint arXiv:2504.12328*.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. 2025. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2506.18896*.

Appendix

Overview. This appendix covers implementation details (§A), the evaluation protocol (§B), additional results and qualitative examples (§C), and discussions on cost and reward hacking (§D).

A Implementation Details

SFT data construction. We randomly select 10k samples and generate one reasoning rollout per sample using the base model. We then filter out samples with reasoning token length larger than 8,192, and use GPT-5 to generate critiques for each reasoning step. Next, we apply a three-stage filter to ensure: (i) structural integrity (reasoning and critique blocks are strictly paired); (ii) answer–critique consistency with the ground-truth critic. This yields **5,056** high-quality samples (about 51% retention). For each retained sample, we interleave reasoning and critique steps to construct a unified trace. The dataset spans a wide range of complexities with token lengths up to 15,130: about 75% of samples fall between 2,048 and 8,192 tokens, 24% exceed 8,192 tokens, and 10% exceed 10,240 tokens, providing sufficient depth for complex multi-step reasoning.

Hyperparameters and lengths. SFT uses batch size 64 for 550 steps. RL with GRPO uses group size G=8, batch size 32, λ_{format}=0.05, and trains for ∼1200 steps total (two-stage schedule below). Maximum response lengths are 8,192 tokens for reasoning and 4,096 tokens for critique. The system prompt is shown in Table 11 (highlighted in orange).

RL training schedule. For RL training, we use a two-stage schedule with different hyperparameter settings. In Stage 1, we set $\lambda_{crit} = 0$, $\lambda_{reason} = 1$, and $\lambda_{format} = 0.05$, and train for about 450 steps to improve reasoning ability and adherence to the reasoning–critique format. In Stage 2, we alternate between two settings every 40 steps for approximately 450 steps: $(\lambda_{reason}, \lambda_{format}, \lambda_{crit}) = (1, 0.05, 0)$ and $(0, 0.05, 1)$. This stage is designed to improve both reasoning ability and critique capability. We alternate these two objectives to mitigate gradient conflict between reasoning and critique optimization.

Full GRPO objective. The GRPO objective maximizes the expected normalized advantage under a KL constraint to the reference policy:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{\sum_k |\tau_k|}\sum_{k=1}^{G}\sum_{t=1}^{|\tau_k|}\Big(\mathcal{L}_{k,t}^{\text{clip}} - \beta\, D_{\text{KL}}\big(\pi_\theta(\tau_k|x) \,\|\, \pi_{\text{ref}}(\tau_k|x)\big)\Big)\Bigg], \tag{9}$$

where the expectation is taken over $(x, a) \sim \mathcal{D}$, $\{\tau_k\}_{k=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, and $\mathcal{L}_{k,t}^{\text{clip}} = \min\{\rho_{k,t}\mathcal{A}_{k,t},\ \text{clip}(\rho_{k,t}, 1-\epsilon, 1+\epsilon)\mathcal{A}_{k,t}\}$; $\rho_{k,t} = \frac{\pi_\theta(\tau_{k,t}|x,\tau_{k,<t})}{\pi_{\theta_{\text{old}}}(\tau_{k,t}|x,\tau_{k,<t})}$; ϵ and β are hyperparameters. Here, similar to DAPO (Yu et al., 2025), we use token-level policy gradient loss to optimize the model, where longer sequences can have more influence.

B Evaluation Protocol Details

We evaluate both the reasoning and critique capability of models. Evaluations are performed with a maximum response length of 8,192 tokens for reasoning and 4,096 tokens for critique by default.

For problem solving capability, we evaluate model reasoning performance using Pass@1 (P@1 for short) and Pass@8, which capture single-attempt accuracy as well as success rates across multiple sampled attempts.

For critique capability, we evaluate models at both the final answer level and the reasoning-step level. i) For the final answer, the correctness of the critique is **directly verifiable** (see discussion for Critique-consistency reward in §3.4.1). ii) For intermediate reasoning steps, however, no human-annotated ground truth exists. Therefore, we employ a strong LLM (GPT-5-mini) as an automatic judge to assess the correctness of the model's stepwise critiques. We denote the pseudo ground-truth critic labels for a trajectory as $z^g = (z_1, z_2, \cdots, z_T)$. Note that we use GPT-5-mini instead of GPT-5 here in considering that GPT-5-mini is much faster than GPT-5 while they achieve similar agreement rate with verifiable critics when evaluated at answer level.

Sampling protocol for critique evaluation. To bound LLM-judge cost (GPT-5-mini step labels are billed per token; per-trajectory labeling cost is dominated by the ∼250 atomic steps in each trajectory), we evaluate critique on a fixed sample of **240 STC trajectories per benchmark**: the first 30 prompts of each dataset × 8 STC rollouts per prompt. This sample is reused for all PRM

comparisons (Table 4) and all critic tables (Tables 2, 3). AIME24 has exactly 30 prompts and is therefore fully covered; AMC23 (83 prompts), MATH-500 (500), Minerva (272), and OlympiadBench (675) are each represented by their first 30 prompts. Reasoning (Pass@1/Pass@8) is evaluated on each benchmark's full prompt set. Extending critique evaluation to full benchmark coverage is a straightforward follow-up bounded only by LLM-judge cost.

Since our work is (to our knowledge) the first to unify stepwise reasoning and critique in a single model, there is no off-the-shelf evaluation metric. Previous PRM benchmarks and evaluation metrics (Zheng et al., 2025) are designed for post-hoc verification and are not suitable for our evaluation since they assume a separate verifier with access to full trajectories and gold step labels, while our setting evaluates a jointly generated step-and-critique process.

Motivated by (Zheng et al., 2025), we design task-aligned metrics that measure correct accuracy (Corr.), error accuracy (Err.), and their harmonic mean (F1).

Correct Accuracy (Corr.). Correct accuracy evaluates critic predictions on instances whose ground-truth critic label is 1 (correct).

At the answer level, this measures the fraction of truly correct final answers that are validated as correct by the critic. Formally, letting s_{T_j} denote the critic's final prediction for sample j and z_{T_j} its label,

$$\text{Corr.} = \frac{\sum_j \mathbf{1}(s_{T_j} = 1 \wedge z_{T_j} = 1)}{\sum_j \mathbf{1}(z_{T_j} = 1)}. \quad (10)$$

Note that each sample may have a different number of steps T_j.

At the process level, this measures the fraction of reasoning steps that are correct according to pseudo ground-truth and are predicted as correct by the critic. Letting $s_t^{(j)}$ denote the critic's prediction at step t for sample j, we have

$$\text{Corr.} = \frac{\sum_j \sum_{t=1}^{T_j} \mathbf{1}(s_t^{(j)} = 1 \wedge z_t^{(j)} = 1)}{\sum_j \sum_{t=1}^{T_j} \mathbf{1}(z_t^{(j)} = 1)}. \quad (11)$$

Error Accuracy (Err.). Similarly, error accuracy evaluates critic predictions on instances whose ground-truth critic label is 0 (incorrect).

At the answer level, this measures the fraction of incorrect final answers that are correctly rejected by the critic. Formally,

$$\text{Err.} = \frac{\sum_j \mathbf{1}(s_{T_j} = 0 \wedge z_{T_j} = 0)}{\sum_j \mathbf{1}(z_{T_j} = 0)}. \quad (12)$$

At the process level, this measures the fraction of reasoning steps that are incorrect according to pseudo ground-truth and are predicted as incorrect by the critic. Formally,

$$\text{Err.} = \frac{\sum_j \sum_{t=1}^{T_j} \mathbf{1}(s_t^{(j)} = 0 \wedge z_t^{(j)} = 0)}{\sum_j \sum_{t=1}^{T_j} \mathbf{1}(z_t^{(j)} = 0)}. \quad (13)$$

F1. Since critique evaluation is inherently imbalanced (correct instances often dominate), we adopt the harmonic mean of correct accuracy and error accuracy for overall critic capability evaluation:

$$\text{Balanced F1} = \frac{2 \cdot \text{Corr.} \cdot \text{Err.}}{\text{Corr.} + \text{Err.}}.$$

This metric penalizes degenerate solutions that achieve high correct accuracy but fail to detect errors, or vice versa.

	GPT-5-mini reference			Claude reference		
Dataset	Corr.	Err.	F1	Corr.	Err.	F1
AIME24	84.4	61.3	71.0	80.1	64.4	**71.4**

Table 5: **STC process-level critic performance under each LLM reference (AIME24).** All values in %. The same STC predictions are scored against either GPT-5-mini or Claude Opus 4.7 step labels. Switching the reference changes Corr. by -4.3% and Err. by $+3.1\%$, with balanced F1 shifting by 0.4% on AIME24.

C Additional Results

C.1 Reliability of GPT-5-mini as a Judge

In the *Reliability of the GPT-5-mini judge* paragraph (§4.2.2) of the main text, we assess the reliability of GPT-5-mini step labels by comparing with verifiable answer-level ground truth (about 90% agreement). Since this is an indirect, answer-level check, we further perform a *direct step-level* validation by re-evaluating the AIME24 labeled corpus with a stronger LLM judge — **Claude Opus 4.7** (with maximum reasoning budget and no output truncation) — under the same per-step prompt as GPT-5-mini (**60k+** step labels in total).

GPT-5-mini achieves balanced F1 of 87.3% (Corr. 88.3%, Err. 86.4%, raw agreement 88.1%) against Claude as gold standard on

Method	Dataset	TP	FP	TN	FN	Valid Ratio	GT Neg. Ratio	Pred. Neg. Ratio
STC-SFT	AIME24	44	17	2	0	26.3	27.0	3.2
	AMC23	321	24	7	5	53.8	8.1	3.4
	MATH	2907	224	18	25	79.4	7.8	1.4
	Minerva	425	777	74	15	59.3	61.4	6.9
	OlympiadBench	1820	754	64	28	49.4	29.3	3.5
	Average	–	–	–	–	53.6	27.0	4.0
STC w/o Critic	AIME24	66	52	18	2	57.5	39.1	14.5
	AMC23	396	88	32	20	80.7	20.1	9.7
	MATH	3223	377	75	87	94.1	12.3	4.3
	Minerva	558	1223	178	32	91.5	63.0	10.5
	OlympiadBench	2190	1535	490	134	80.5	38.4	14.3
	Average	–	–	–	–	80.9	35.0	11.0
STC	AIME24	55	23	35	12	52.1	28.0	37.6
	AMC23	363	48	59	39	76.7	17.1	19.3
	MATH	3188	308	149	127	94.3	11.5	7.3
	Minerva	545	1053	295	60	89.8	57.0	18.2
	OlympiadBench	2084	1028	777	218	76.1	30.3	24.2
	Average	–	–	–	–	77.8	29.0	21.0

Table 6: **Answer-level confusion and statistics.** All ratios in %. Valid Ratio: fraction of samples where the final critic could be extracted (failures usually due to token overlength or unmatched `<score></score>`); we only count valid samples. *GT Neg. Ratio* is the ground-truth negative (z=0) rate; *Pred. Neg. Ratio* is the predicted negative rate. The base model DS-Qwen-1.5B has 78.4% average valid ratio (due to over-length).

Method	Dataset	TP	FP	TN	FN	GT Neg. Ratio	Pred. Neg. Ratio
SFT	AIME24	38654	7471	4102	2644	21.9	12.8
	AMC23	31686	4487	2049	1862	16.3	9.8
	MATH	19737	2228	713	559	12.7	5.5
	Minerva	29822	8242	1976	1455	24.6	8.3
	OlympiadBench	32043	6818	5567	3322	25.9	18.6
	Average	30388	5849	2881	1968	20.3	11.0
STC w/o Critic	AIME24	40348	6438	3496	2568	18.8	11.5
	AMC23	28724	4469	1918	1577	17.4	9.5
	MATH	17185	1656	552	694	11.0	6.2
	Minerva	24808	6705	1715	1424	24.3	9.1
	OlympiadBench	32281	5956	5335	3628	23.9	19.0
	Average	28669	5045	2603	1978	19.1	11.1
STC	AIME24	41205	4471	7082	7612	19.1	24.3
	AMC23	29572	3098	4558	5652	17.9	23.8
	MATH	15815	1178	1343	1809	12.5	15.7
	Minerva	25370	4661	2772	2948	20.8	16.0
	OlympiadBench	32718	3496	10634	9261	25.2	35.5
	Average	28936	3381	5278	5456	19.1	23.1

Table 7: **Process-level confusion and statistics.** Computed on the first 240 trajectories (30 prompts $\times$ 8 rollouts) per benchmark.

AIME24. The disagreement is asymmetric: GPT-5-mini calls 19.2% of steps wrong whereas Claude calls only 10.1%, *i.e.*, GPT-5-mini is a more conservative judge that over-flags errors by roughly $1.9\times$, but it almost never *misses* a Claude-flagged error.

STC critic performance under two judges. Table 5 re-computes STC's process-level critic performance on AIME24 using *either* GPT-5-mini or Claude as the reference. On AIME24, balanced F1 differs by only 0.4% ($71.0\% \rightarrow 71.4\%$). The two metrics move in opposite directions when the reference is swapped: Corr. drops 4.3% (Claude's larger correct-class makes a stricter denominator) while Err. rises 3.1% (Claude's wrong-class is smaller and cleaner).

C.2 More Statistics for Critiques

Table 6 and Table 7 show the answer-level and step-level confusion matrixes and statistics.

In Table 6, to ensure transparency, we additionally report the valid ratio (VR), *i.e.*, the proportion of samples feasible for evaluation of the critique performance. In general, the failures in extracting final critic are treated as invalid samples, which are usually due to token overlength or unmatched `<score></score>`. Different schemes present different valid ratios. Since there are large differences on the valid ratios between *STC-SFT* and the other two schemes, the two schemes *STC w/o Critic* and

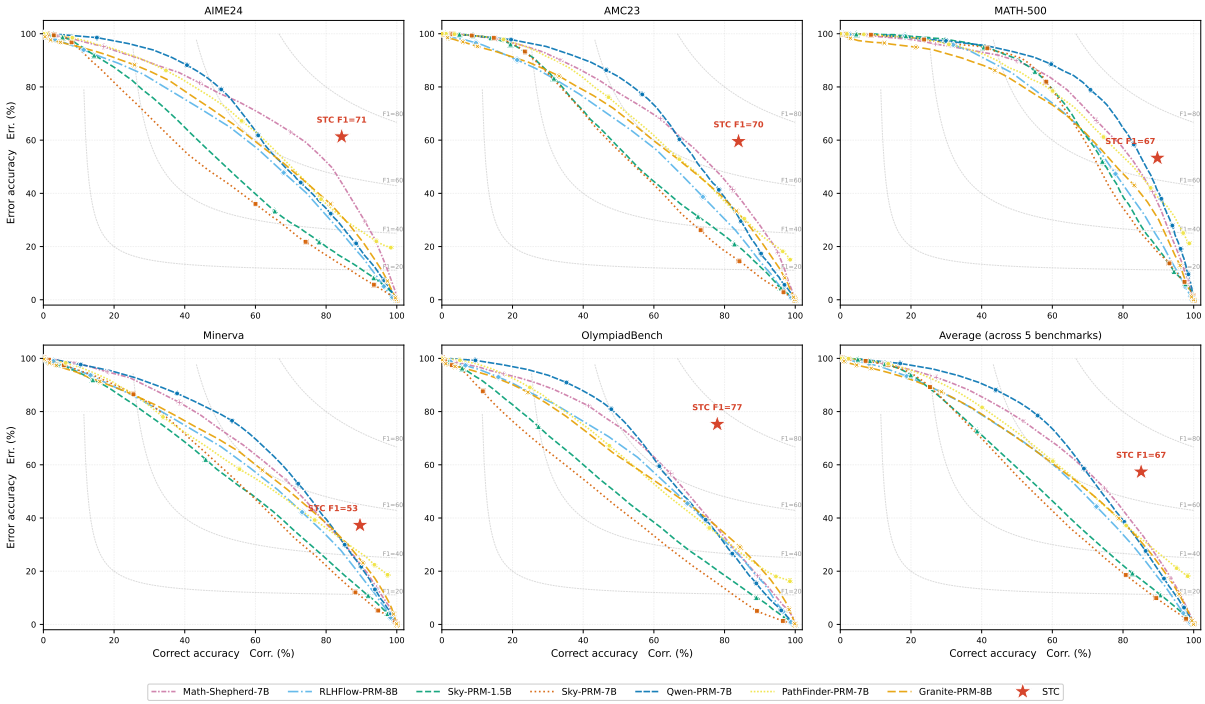


Figure 4: **Step-level Corr. vs. Err. trade-off across 5 benchmarks.** STC (★, single calibration-free point with annotated F1) vs. seven external PRMs (curves swept over ∼200 thresholds in $[0.001, 0.999]$). The *Average* subplot is the cross-dataset arithmetic mean. *Legend "Qwen-PRM-7B" refers to Qwen2.5-Math-PRM-7B.*

STC with similar valid ratios are more fairly comparable to analyse the performance.

Since critique performance is computed only on valid samples, VR contextualizes comparability across methods.

For step-level evaluation, since each critique is following a reasoning step, the reasoning-critique pairs even for over-length traces are also used for evaluation. Therefore we do not calculate the valid ratio.

C.3 More Comparisons of STC and External PRMs

We complement Table 4 with two additional comparison axes: (i) full Corr.–Err. trade-off curves under threshold sweep, and (ii) inference compute and memory.

Accuracy: Corr. vs. Err. trade-off curves. Figures 4 and 5 show each PRM's full Corr.–Err. trade-off under threshold sweep (on 30 prompts × 8 rollouts = 240 trajectories), at the step level and answer level respectively. The seven PRMs span training context lengths of 4K–128K; positions beyond the trained context are handled via RoPE extrapolation.

Setup	Params	FLOPs	Mem	STC saves	
	(B)	(T)	(GiB)	FLOPs	Mem
STC (single 1.5B)	1.5	26.0	3.0	—	—
1.5B + Skywork-PRM-1.5B	3.0	36.1	5.9	28%	49%
1.5B + 7B PRM	8.5	102.3	16.3	75%	82%
1.5B + 8B PRM	9.5	114.3	18.6	77%	84%

Table 8: **Per-trajectory inference cost: STC vs separate-verifier PRM pipelines.** FLOPs computed as $2N_{\text{params}}T$ with measured R=6,016 reasoning tokens and C=2,638 critique tokens on AIME24. Peak memory is fp16 weights plus GQA KV cache[2] for an 8K-token trajectory at batch size 1, assuming concurrent serving (both reasoner and PRM resident on the GPU). 7B PRMs: Math-Shepherd, RLHFlow, Qwen2.5-Math-PRM, PathFinder. 8B PRM: Granite-3.3.

Step level. STC's single calibration-free point sits on or above every PRM's full sweep on AIME24, AMC23, and OlympiadBench, and is competitive on MATH-500 and Minerva. In the *Average* panel, STC reaches the best step-F1 in expectation.

Answer level. STC's point lies on the PRM frontier on most datasets (four out of five), aligned with the best-performing PRM curves. The answer-level

[2]The KV cache is <8% of total memory in every configuration because all baselines use grouped-query attention (e.g., 2 KV heads for the 1.5B model), so peak memory is dominated by model weights.

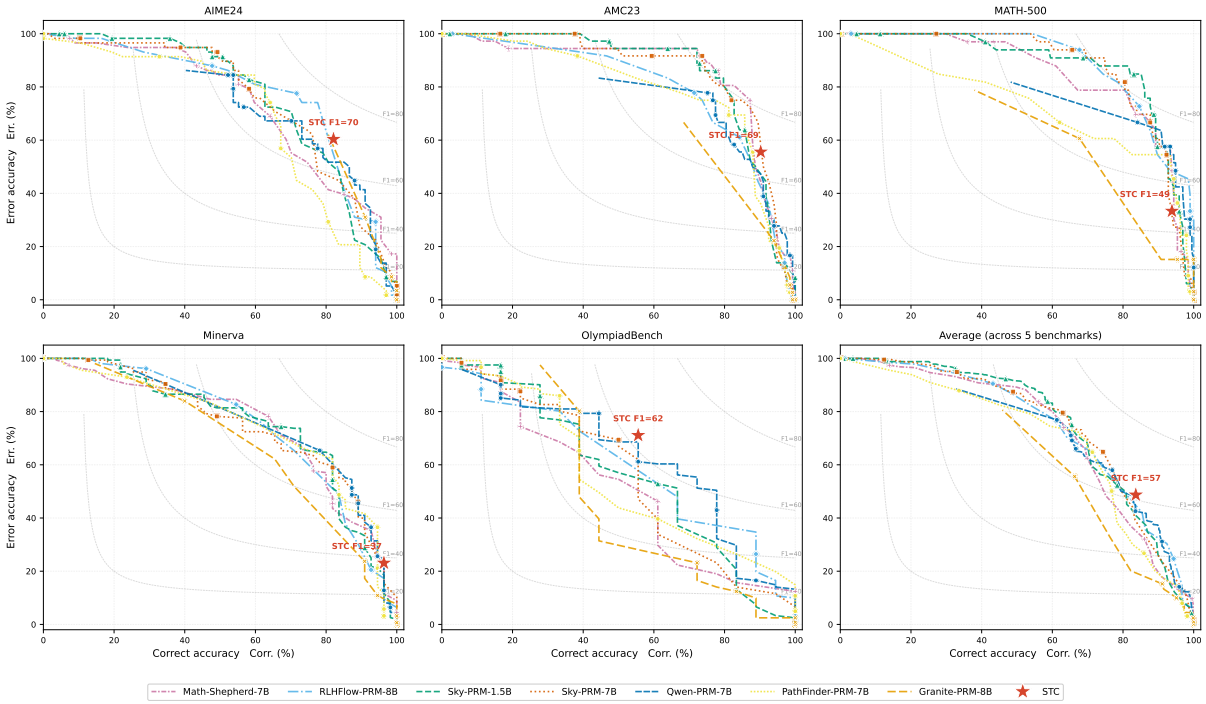


Figure 5: **Answer-level Corr. vs. Err. trade-off across 5 benchmarks.** Same protocol as Figure 4.

curves are visibly less smooth than the step-level ones because each curve is now built from only 240 trajectory outcomes per benchmark, versus tens of thousands of step labels at the step level (Table 7).

Inference compute and memory. A separate-verifier pipeline must load and run two models: a reasoner (N_R params) generates R reasoning tokens autoregressively, then a PRM (N_{PRM} params) prefills the trajectory to score each step, costing approximately $2R(N_R{+}N_{PRM})$ FLOPs. STC instead emits R reasoning tokens and C critique tokens inline within a single 1.5B model, costing $\approx 2N_R(R{+}C)$ FLOPs. Empirically, on the 240 STC trajectories from AIME24 (30 prompts $\times$ 8 rollouts), $C/R \approx 0.44$. Table 8 summarizes the resulting per-trajectory cost: STC saves $\sim$28% FLOPs and $\sim$49% peak memory against the smallest 1.5B-PRM baseline, and $\sim$75–77% FLOPs and $\sim$82–84% peak memory against the 7B/8B baselines. STC additionally avoids the system overhead of dispatching trajectories between two models (scheduling and cross-model serialization).

C.4 Pass@k Variability Across Evaluation Seeds

AIME24 (30 problems) and AMC23 (83 problems) are small benchmarks of the five testing datasets; a single $n{=}8$ rollout evaluation can be noisy. In contrast, MATH-500, Minerva, and OlympiadBench contain 500, 272, and 675 problems respectively, so their single-seed estimates are much more robust. To quantify the small-benchmark noise, we extend the rollout budget to $n{=}128$ per prompt for the STC (full) checkpoint on AIME24 and AMC23, and partition the 128 rollouts into 16 disjoint 8-rollout splits, treated as 16 independent eval seeds. Table 9 compares the original single-seed result (the value reported in Table 1) against the 16-seed mean$\pm$std, where std is the spread of a single 8-rollout evaluation across the 16 seeds. The std quantifies how far an arbitrary single-seed evaluation may drift.

Eval setup	AIME24		AMC23	
	P@1	P@8	P@1	P@8
1 seed	27.9	56.7	60.5	81.9
16 seeds (mean$\pm$std)	27.7 ± 2.0	56.5 ± 5.5	62.3 ± 0.9	83.7 ± 1.8

Table 9: **Pass@1 / Pass@8 (%) variability of STC (full) on AIME24 and AMC23, measured by mean$\pm$std across 16 disjoint 8-rollout seeds** ($N{=}128$ **rollouts/prompt).** The 1-seed row reproduces values from Table 1; the 16-seed row uses the same checkpoint.

On AIME24, the 16-seed mean almost coincides with the single-seed value (shifts within $\sim$0.2 %, negligible against the single-seed std of $\sim$2–6 %). On AMC23, the 16-seed mean is about 1.8 %

Model	AIME24		AMC23		MATH-500		Minerva		OlympiadBench		Average	
	P@1	P@8	P@1	P@8	P@1	P@8	P@1	P@8	P@1	P@8	P@1	P@8
STC	27.9	**56.7**	60.5	81.9	**82.9**	**92.6**	**27.9**	**49.3**	42.7	57.3	48.4	**67.6**
STC w/ Dense	**30.4**	53.3	**64.2**	**83.1**	82.6	92.0	27.3	48.2	**43.6**	**59.0**	**49.6**	67.1

Table 10: **Influence of stepwise critique as dense reward.** Reasoning performance (%) on five math benchmarks, evaluated with Pass@1 (P@1) and Pass@8 (P@8). **Bold** marks the better of the two per column.

higher than the single seed results reported in Table 1.

C.5 Discussion on Stepwise Critique as Dense Rewards

Alternatively, we leverage stepwise critique judgments as dense rewards to further improve learning of the reasoning process.

Concretely, the critique model produces a binary decision $s_n^{(k)} \in \{0, 1\}$ at each reasoning step n of trajectory τ_k, indicating whether the reasoning step is correct or not. We obtain step-level advantages by summing the normalized step rewards of its following steps:

$$A_k^{dense,n} = \sum_{n' \geq n} \frac{s_{n'}^{(k)} - \text{mean}(\mathcal{S})}{\text{std}(\mathcal{S})}, \tag{14}$$

where $\mathcal{S} = \{s_n^{(k)} | n = 1, 2, \cdots, T^{(k)}, k = 1, 2, \cdots, G\}$ denotes the set including all step critique rewards within the group.

The stepwise reasoning advantage for the t-th token is

$$\mathcal{A}_{k,t}^{\text{dense}} = A_k^{\text{dense},s(t)}, \tag{15}$$

where $s(t)$ denotes the identification of the reasoning step index that the t-th token belongs to.

The overall advantage for the t-th token of the trajectory τ_k is then augmented by this dense critique signal as:

$$\widetilde{\mathcal{A}_{k,t}} = \mathcal{A}_{k,t} + \lambda_{\text{dense}} \mathcal{A}_{k,t}^{\text{dense}}, \tag{16}$$

where λ_{dense} balances the contribution of dense stepwise feedback ($\lambda_{\text{dense}} = 0.5$).

This extension transforms the optimization from relying solely on sparse trajectory-level rewards to incorporating fine-grained, step-level supervision. By providing dense reward shaping, the model receives immediate reward signals for fine-grained optimization.

Finally, the GRPO objective maximizes the expected normalized advantage under a KL constraint to the reference policy:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{\sum_k |\tau_k|} \sum_{k=1}^{G} \sum_{t=1}^{|\tau_k|} \Big(\widetilde{\mathcal{L}}_{k,t}^{\text{clip}} - \beta\, D_{\text{KL}}\big(\pi_\theta(\tau_k|x) \,\|\, \pi_{\text{ref}}(\tau_k|x)\big)\Big)\Bigg], \tag{17}$$

where the expectation is taken over $(x, a) \sim \mathcal{D}$, $\{\tau_k\}_{k=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$, and $\widetilde{\mathcal{L}}_{k,t}^{\text{clip}} = \min\{\rho_{k,t}\widetilde{\mathcal{A}_{k,t}},\ \text{clip}(\rho_{k,t}, 1-\epsilon, 1+\epsilon)\widetilde{\mathcal{A}_{k,t}}\}$; $\rho_{k,t} = \frac{\pi_\theta(\tau_{k,t}|x,\tau_{k,<t})}{\pi_{\theta_{\text{old}}}(\tau_{k,t}|x,\tau_{k,<t})}$; ϵ and β are hyperparameters.

We implemented this and Table 10 shows the results. Compared with plain STC, dense reward supervision shifts average Pass@1 by $+1.2\%$ and average Pass@8 by -0.5%.

We thus thought our existing implementation does not achieve a robust gain. More effective reward shaping may be needed to unlock the potential of stepwise critique as a dense reward, which we leave as future work.

C.6 More Qualitative Examples

Tables 11 to 14 present four full-mode trajectories spanning four benchmarks (AIME24, Minerva, MATH-500, OlympiadBench), illustrating the qualitative behaviors that STC's critic exhibits in practice.

Across these examples the critique is concise and informative: it pinpoints the *cause* of a detected error rather than emitting a binary flag. The four examples illustrate, but are not limited to, the following error types that STC successfully identifies: (i) **Invalid mathematical shortcut** – Table 11 (AIME24, lottery probability): STC rejects a bogus alternative formula $1/\binom{6}{2}$ that the model briefly floats, pointing back at the prior "given at least two" enumeration; the trajectory re-derives the correct answer $1/115$. (ii) **Meta-reasoning over irrelevant context** – Table 12 (Minerva): STC identifies that the interferometer description is irrelevant to a *single*-dish-resolution question. (iii) **Constraint misreading** – Table 13 (MATH-500, Greek army combinations): STC catches that the army must

field *exactly* the demanded force, not "meet or exceed" it. (iv) **Factual and symbolic errors in long trajectories** – Table 14 (OlympiadBench, prime y_n sequence, 188 steps): STC catches two independent errors, a wrong uniqueness claim about $a{=}1$ and a wrong expression for y_2. Color conventions: critiques identified as correct steps are shown in green, critiques identified as incorrect steps in red, and the system prompt in orange. For readability we display the first and last few steps verbatim, plus a ± 1-step window around each `<score>0</score>`, and skip the (uniformly correct) intermediate steps.

C.7 Compact Mode

Tables 15 to 18 show the four problems from §C.6 re-run in compact mode (system prompt stripped of `<critic>`/`<score>` instructions), so the model emits reasoning only. The two modes differ *only* in the system prompt; the underlying STC checkpoint and decoding parameters are identical.

C.8 Failure Case

Following §4.2.5, Table 19 shows a failure case where STC's critic itself is unreliable: at step 3 it confirms a wrong unit conversion ("3 Mpc equals 3000 parsecs") with `<score>1</score>`. The pseudo-GT GPT-5-mini judge identifies "*Incorrect unit conversion:* $1\,\mathrm{Mpc} = 10^6\,\mathrm{pc}$, *so* $3\,\mathrm{Mpc} = 3{,}000{,}000\,\mathrm{pc}$, *not* $3{,}000\,\mathrm{pc}$." This example shows that, while STC's natural-language critique offers step-level interpretability, the critic itself is not 100% reliable; further improving its performance remains a direction for future work.

D Additional Discussion

D.1 Computation Cost

We validate STC on a 1.5B model. We set the maximum response length as 8,192 tokens for reasoning and 4,096 tokens for critique. Running 1200 RL steps requires approximately 5 days on 16×40GB A100 GPUs.

Our goal is to establish the core mechanism; the framework is model-agnostic and extendable to other models. Due to limited resources, we did not extend the validation to larger scale or multimodal experiments. We hope this work motivates further research on critical-thinking mechanisms.

D.2 About Reward Hacking

One may wonder whether our scheme faces reward hacking. Actually, we did not observe evidence of reward hacking in extensive inspection.

First, as shown in Table 6, STC identifies 21% of final answers as incorrect, rather than collapsing to predicting all answers as either correct or incorrect. Similarly, STC identifies 23% of steps as incorrect on average, close to the pseudo ground-truth ratio of 19.1% (see Table 7).

Second, as shown in Table 11, our critic correctly flags a bogus alternative computation that the model floats mid-trace, providing valuable information rather than merely stylistic output.

D.3 Inference Cost

The full mode interleaves critiques with reasoning and consumes additional tokens at inference time. Naive deployment without exposing the compact-mode fallback could increase cost, latency, and energy use in user-facing applications. We recommend defaulting to compact mode and enabling full mode only when interpretability is required.

Stepwise Think-Critique (Full Mode) – AIME24 example

<|User|>: Jen enters a lottery by picking 4 distinct numbers from $S = \{1, 2, 3, \cdots, 9, 10\}$. 4 numbers are randomly chosen from S. She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers. The probability of her winning the grand prize given that she won a prize i...

Let's think step by step and output the final answer within \\boxed{}.
Leverage the output `\n\n` as the indicator for separating reasoning steps. Following each reasoning step, please provide a critique and a score, where the critique reasons about the correctness of this step within `<critic>` and `</critic>` tags, the score of 1 for denoting correct or 0 denoting incorrect within `<score>` and `</score>` tags. Follow the format in this example precisely: [Your thinking process for a step goes here] `<critic>` [Your brief justification for the score goes here] `</critic>` `<score>` [0 or 1] `</score>`

<|Assistant|><think>

Step 1. Alright, so I've got this probability problem here, and I need to figure it out step by step. Let me try to understand what's being asked.
<critic>Clear setup and understanding of the problem context.</critic> <score>1</score>[GPT-5-mini: 1]

Step 2. Jen is entering a lottery by picking 4 distinct numbers from the set S = 1, 2, 3, ..., 9, 10. That means S has 10 numbers, right? So, the total number of possible ways she can pick her 4 numbers is the combination of 10 taken 4 at a time. I remember that the formula for combinations is C(n, k)...
<critic>Correct identification of total combinations C(10,4).</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 27 thinking steps . . .)

Step 30. Yes, that seems correct.
<critic>Affirmation of correctness; consistent with prior results.</critic> <score>1</score>[GPT-5-mini: 1]

Step 31. Alternatively, another way to think about it is that given that at least two numbers are correct, the probability that all four are correct is 1 / C(6, 2). Wait, no, that might not be the right way.
<critic>The suggested alternative (1/C(6,2)) is not valid; the "given at least two" reasoning is fine.</critic> <score>0</score>[GPT-5-mini: 0]

Step 32. Wait, perhaps I can compute it as the ratio of the number of ways to have exactly 4 correct over the number of ways to have at least 2 correct.
<critic>Correct ratio interpretation.</critic> <score>1</score>[GPT-5-mini: 1]

Step 33. So, the number of ways to have exactly 4 correct is 1, as before. The number of ways to have exactly 2 correct is C(4,2)*C(6,2) = 90, and exactly 3 is C(4,3)*C(6,1)=24. So, total is 115. So, the ratio is 1/115.
<critic>Correct recalculation confirming 115 and ratio.</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 13 thinking steps . . .)

Step 47. So, same result.
<critic>Consistent result noted.</critic> <score>1</score>[GPT-5-mini: 1]

Step 48. Therefore, I think that's correct.
<critic>Conclusion is justified.</critic> <score>1</score>[GPT-5-mini: 1]

Step 49. So, the probability is 1/115, so m + n is 1 + 115 = 116.
<critic>Correct restatement of result.</critic> <score>1</score>[GPT-5-mini: 1]

Step 50. **Final Answer** $\boxed{116}$
</think>
<critic>Final answer is correct.</critic> <score>1</score>[GPT-5-mini: 1]

Step 51. Jen enters a lottery by picking 4 distinct numbers from the set $S = \{1, 2, 3, \ldots, 9, 10\}$. The total number of possible ways to pick her 4 numbers is given by the combination $\binom{10}{4} = 210$.
<critic>Correct restatement of setup and total combinations.</critic> <score>1</score>[GPT-5-mini: 1]

Step 52. To win a prize, she needs at least two of her numbers to be among the four randomly chosen numbers. The probability of winning the grand prize is when all four of her numbers are the randomly chosen numbers.
<critic>Accurate description of winning conditions.</critic> <score>1</score>[GPT-5-mini: 1]

Step 53. First, we calculate the probability of winning the grand prize. There is only one favorable outcome (matching all four numbers), so the probability is: $P(A) = \frac{1}{210}$
<critic>Correct computation of P(A).</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 4 summary blocks . . .)

Step 58. The conditional probability of winning the grand prize given that she won a prize is: $P(A|B) = \frac{P(A)}{P(B)} = \frac{\frac{1}{210}}{\frac{115}{210}} = \frac{1}{115}$
<critic>Correct application of conditional probability.</critic> <score>1</score>[GPT-5-mini: 1]

Step 59. Therefore, the probability is $\frac{1}{115}$, and the sum of m and n where the probability is $\frac{m}{n}$ is: $m + n = 1 + 115 = 116$
<critic>Correct final probability and sum.</critic> <score>1</score>[GPT-5-mini: 1]

Step 60. $\boxed{116}$
<critic>Correct final boxed result.</critic> <score>1</score>[GPT-5-mini: 1]

Table 11: **Qualitative example (AIME24, lottery probability).** A typical STC trajectory with paired critiques. At step 31 STC catches the model floating a bogus alternative computation ($\Pr(\text{all four correct} \mid \text{at least two correct}) = 1/\binom{6}{2}$) and notes that the preceding 'given at least two' reasoning is the correct path; at steps 32–33 the model returns to the correct enumeration and re-derives the answer $1/115$. The final answer matches the ground truth. Similar to our base model DS-Qwen-1.5B, the post-`</think>` blocks are STC's brief summary restatement. The gray "Step N" labels are added for display only; the model emits steps separated by `\n\n` without numeric IDs. We also leverage gray `[GPT-5-mini: 0/1]` tag to show the pseudo-GT judge's verdict.

Stepwise Think-Critique (Full Mode) – Minerva example

<|User|>: Preamble: A radio interferometer, operating at a wavelength of $1\ \mathrm{cm}$, consists of 100 small dishes, each $1\ \mathrm{m}$ in diameter, distributed randomly within a $1\ \mathrm{km}$ diameter circle. What is the angular resolution of a single dish, in radians?

Let's think step by step and output the final answer within \\boxed{}.
Leverage the output \n\n as the indicator for separating reasoning steps. Following each reasoning step, please provide a critique and a score, where the critique reasons about the correctness of this step within <critic> and </critic> tags, the score of 1 for denoting correct or 0 denoting incorrect within <score> and </score> tags. Follow the format in this example precisely: [Your thinking process for a step goes here] <critic> [Your brief justification for the score goes here] </critic> <score> [0 or 1] </score>

<|Assistant|><think>

Step 1. Okay, so I have this problem about a radio interferometer and I need to find the angular resolution of a single dish. Hmm, let me start by recalling what angular resolution means. I think it's related to how well the telescope can distinguish between two closely spaced points in the sky. The smal...
<critic>Correct understanding of angular resolution and its purpose.</critic> <score>1</score>[GPT-5-mini: 1]

Step 2. The problem says it's a radio interferometer with 100 small dishes, each 1 meter in diameter, distributed randomly within a 1 km diameter circle. The wavelength operating is 1 cm. Wait, so it's a single dish, not an interferometer. Hmm, okay, so maybe I don't need the interferometer part for this...
<critic>Confuses the task; the interferometer data is irrelevant to single-dish resolution.</critic> <score>0</score>[GPT-5-mini: 0]

Step 3. So, the formula for angular resolution in radio telescopes is similar to the optical telescopes, right? It's the diffraction limit, which is given by the wavelength divided by the diameter of the telescope. So, the formula is $\theta = \lambda$ / D, where θ is the angular resolution in r...
<critic>Correct statement of the Rayleigh formula for resolution.</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 12 thinking steps where STC critique matches GT . . .)

Step 16. 1 cm is 0.01 meters, and 1 m is 1 meter. So, 0.01 m / 1 m = 0.01 radians. So, yes, that's correct.
<critic>Correct recheck of units and computation.</critic> <score>1</score>[GPT-5-mini: 1]

Step 17. Alternatively, 1 cm is 0.01 m, so 0.01 m divided by 1 m is 0.01. So, 0.01 radians.
<critic>Correct reaffirmation of the computation.</critic> <score>1</score>[GPT-5-mini: 1]

Step 18. So, I think that's the answer. So, the angular resolution is 0.01 radians.
<critic>Correct conclusion.</critic> <score>1</score>[GPT-5-mini: 1]

Step 19. **Final Answer** The angular resolution of a single dish is $\boxed{0.01}$ radians.
</think>
<critic>Correct final answer.</critic> <score>1</score>[GPT-5-mini: 1]

Step 20. To find the angular resolution of a single dish, we use the formula for angular resolution, which is given by the wavelength divided by the diameter of the telescope. The formula is:
<critic>Accurate description of the method.</critic> <score>1</score>[GPT-5-mini: 1]

Step 21. $\theta = \frac{\lambda}{D}$
<critic>Correct formula.</critic> <score>1</score>[GPT-5-mini: 1]

Step 22. Given: - Wavelength, $\lambda = 1\ \mathrm{cm} = 0.01\ \mathrm{m}$ - Diameter, $D = 1\ \mathrm{m}$
<critic>Correct conversion and values.</critic> <score>1</score>[GPT-5-mini: 1]

Step 23. Substituting the given values into the formula:
<critic>Appropriate transition to substitution.</critic> <score>1</score>[GPT-5-mini: 1]

Step 24. $\theta = \frac{0.01\ \mathrm{m}}{1\ \mathrm{m}} = 0.01$ radians
<critic>Correct substitution and result.</critic> <score>1</score>[GPT-5-mini: 1]

Step 25. Thus, the angular resolution of a single dish is:
<critic>Correct lead-in to conclusion.</critic> <score>1</score>[GPT-5-mini: 1]

Step 26. $\boxed{0.01}$
<critic>Correct final statement of the value.</critic> <score>1</score>[GPT-5-mini: 1]

Table 12: **Qualitative example (Minerva, single-dish angular resolution).** STC catches a meta-reasoning error at step 2: the trajectory begins by treating an interferometer description as relevant when the question only asks about a single dish. STC's critic correctly flags this misreading. The final answer matches the ground truth.

Stepwise Think-Critique (Full Mode) – MATH-500 example

<|User|>: The Greek army contained two types of soldiers: the upper class and the lower class soldiers. If there were a total of 5 upper class soldiers, and 10 lower class soldiers in a certain part of Athens, and the battle of Thermopylae demands a force of 4 upper class soldiers and 8 lower class soldiers, how many different battalions can be sent?

Let's think step by step and output the final answer within \\boxed{}.
Leverage the output \n\n as the indicator for separating reasoning steps. Following each reasoning step, please provide a critique and a score, where the critique reasons about the correctness of this step within <critic> and </critic> tags, the score of 1 for denoting correct or 0 denoting incorrect within <score> and </score> tags. Follow the format in this example precisely: [Your thinking process for a step goes here] <critic> [Your brief justification for the score goes here] </critic> <score> [0 or 1] </score>

<|Assistant|><think>

Step 1. Okay, so I need to figure out how many different battalions can be sent based on the given information. Let me read the problem again and break it down step by step.
<critic>Good problem setup; no incorrect claims.</critic> <score>1</score>[GPT-5-mini: 1]

Step 2. First, the problem mentions that there are two types of soldiers in the Greek army: upper class and lower class. In a certain part of Athens, there are 5 upper class soldiers and 10 lower class soldiers. So, that's 5 upper and 10 lower soldiers in total.
<critic>Correct identification of given quantities.</critic> <score>1</score>[GPT-5-mini: 1]

Step 3. Then, the battle of Thermopylae demands a force of 4 upper class soldiers and 8 lower class soldiers. So, the Greek army needs to send a combination of upper and lower class soldiers that meets or exceeds the demand of 4 upper and 8 lower soldiers.
<critic>Incorrect requirement; the force must not exceed demand.</critic> <score>0</score>[GPT-5-mini: 0]

Step 4. The question is asking how many different battalions can be sent. I think this means how many different ways we can choose 4 upper class soldiers from the 5 available and 8 lower class soldiers from the 10 available. So, it's a combinatorics problem where we need to calculate the number of combin...
<critic>Correct interpretation of combinations and multiplication principle.</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 18 thinking steps where STC critique matches GT . . .)

Step 23. Wait, let me just make sure I didn't make any mistakes here. So, 5 upper class soldiers, choosing 4: that's 5 ways. 10 lower class soldiers, choosing 8: that's 45 ways. Multiply them together to get the total number of combinations, which is 5 * 45 = 225.
<critic>Accurate verification of prior computations.</critic> <score>1</score>[GPT-5-mini: 1]

Step 24. Yes, that seems correct. I don't think I missed anything here. The problem is about combinations, not permutations, since the order of soldiers in a battalion doesn't matter. So, we're just selecting groups, not arranging them in a specific order.
<critic>Correct clarification about combinations vs permutations.</critic> <score>1</score>[GPT-5-mini: 1]

Step 25. Therefore, I think 225 is the correct answer.
<critic>Correct reaffirmation of result.</critic> <score>1</score>[GPT-5-mini: 1]

Step 26. **Final Answer** The total number of different battalions that can be sent is $\boxed{225}$.
</think>
<critic>Correct final statement of the answer.</critic> <score>1</score>[GPT-5-mini: 1]

Step 27. The problem requires determining the number of different battalions that can be sent given the constraints of the battle of Thermopylae. The Greek army has 5 upper class soldiers and 10 lower class soldiers. The required force is 4 upper class soldiers and 8 lower class soldiers.
<critic>Accurate problem restatement.</critic> <score>1</score>[GPT-5-mini: 1]

Step 28. To solve this, we need to calculate the number of ways to choose 4 upper class soldiers from 5 and 8 lower class soldiers from 10. This is a combinatorics problem involving combinations.
<critic>Correct identification of required combinations.</critic> <score>1</score>[GPT-5-mini: 1]

Step 29. First, we calculate the number of ways to choose 4 upper class soldiers from 5: $C(5,4) = \frac{5!}{4!(5-4)!} = \frac{5!}{4!\cdot 1!} = \frac{120}{24\cdot 1} = 5$
<critic>Correct calculation of C(5,4).</critic> <score>1</score>[GPT-5-mini: 1]

Step 30. Next, we calculate the number of ways to choose 8 lower class soldiers from 10. Using the combination formula and the symmetry property of combinations:
$C(10,8) = C(10,2) = \frac{10!}{2!(10-2)!} = \frac{10\cdot 9\cdot 8!}{2\cdot 1\cdot 8!} = \frac{90}{2} = 45$
<critic>Correct calculation of C(10,8) via symmetry.</critic> <score>1</score>[GPT-5-mini: 1]

Step 31. Multiplying the number of ways to choose the upper class soldiers by the number of ways to choose the lower class soldiers gives the total number of different battalions:
$5 \times 45 = 225$
<critic>Correct multiplication yielding total combinations.</critic> <score>1</score>[GPT-5-mini: 1]

Step 32. Thus, the total number of different battalions that can be sent is $\boxed{225}$.
<critic>Correct concluding statement.</critic> <score>1</score>[GPT-5-mini: 1]

Table 13: **Qualitative example (MATH-500, Greek army combinations).** At step 3 STC catches a misreading of the problem constraint (the army must field exactly the demanded force, not "meet or exceed" it). The final answer matches the ground truth.

Stepwise Think-Critique (Full Mode) – OlympiadBench example

<|User|>: For a positive integer a, define a sequence of integers $x_1, x_2, \ldots$ by letting $x_1 = a$ and $x_{n+1} = 2x_n + 1$ for $n \geq 1$. Let $y_n = 2^{x_n} - 1$. Determine the largest possible k such that, for some positive integer a, the numbers $y_1, \ldots, y_k$ are all prime.

Let's think step by step and output the final answer within \\boxed{}.
Leverage the output \n\n as the indicator for separating reasoning steps. Following each reasoning step, please provide a critique and a score, where the critique reasons about the correctness of this step within <critic> and </critic> tags, the score of 1 for denoting correct or 0 denoting incorrect within <score> and </score> tags. Follow the format in this example precisely: [Your thinking process for a step goes here] <critic> [Your brief justification for the score goes here] </critic> <score> [0 or 1] </score>

<|Assistant|><think>

Step 1. Alright, so I've got this problem here. Let me try to understand it step by step. It says:
<critic>Introductory setup with no incorrect claims.</critic> <score>1</score>[GPT-5-mini: 1]

Step 2. "For a positive integer a, define a sequence of integers $x_1, x_2, \ldots$ by letting $x_1 = a$ and $x_{n+1} = 2x_n + 1$ for $n \geq 1$. Let $y_n = 2^{x_n} - 1$. Determine the largest possible k such that, for some positive integer a, the numbers $y_1, \ldots, y_k$ are all...
<critic>Accurate restatement of the problem.</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 71 thinking steps where STC critique matches GT . . .)

Step 74. $x_1 = 10$, $y_1 = 2^{10} - 1 = 1024 - 1 = 1023$, which is 3*11*31, composite.
<critic>Correct factorization of 1023.</critic> <score>1</score>[GPT-5-mini: 1]

Step 75. Wait, maybe $a = 1$ is the only one that gives y_1 prime, but y_2 is not.
<critic>Incorrect claim about a = 1 giving only y1 prime.</critic> <score>0</score>[GPT-5-mini: 0]

Step 76. Wait, so $a = 2$ and $a = 3$ give $k = 2$. Let's see if there's an a such that y_1 and y_2 are both prime.
<critic>Valid idea to look for a with both y1 and y2 prime.</critic> <score>1</score>[GPT-5-mini: 1]

Step 77. So, $y_1 = 2^a - 1$ must be prime, and $y_2 = 2^{(a+1)\cdot 2^0 - 1} - 1 = 2^{a+1-1} - 1 = 2^a - 1$. Wait, no, hold on.
<critic>Incorrect assertion about y2 expression.</critic> <score>0</score>[GPT-5-mini: 0]

Step 78. Wait, no, earlier we had $x_n = (a+1) \cdot 2^{n-1} - 1$. So, for $n = 2$, $x_2 = (a+1) \cdot 2^1 - 1 = 2(a+1) - 1 = 2a + 2 - 1 = 2a + 1$. So, $y_2 = 2^{2a+1} - 1$.
<critic>Corrects and computes x2 and y2 correctly.</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 98 thinking steps where STC critique matches GT . . .)

Step 177. So, the maximum k is 2, achieved by $a = 2$ and $a = 3$.
<critic>Correct conclusion.</critic> <score>1</score>[GPT-5-mini: 1]

Step 178. Therefore, the answer is $\boxed{2}$.
<critic>Correct final answer.</critic> <score>1</score>[GPT-5-mini: 1]

Step 179. **Final Answer** $\boxed{2}$
</think>
<critic>Final answer is correct.</critic> <score>1</score>[GPT-5-mini: 1]

Step 180. To determine the largest possible k such that, for some positive integer a, the numbers $y_1, \ldots, y_k$ are all prime, we start by analyzing the given sequences.
<critic>Accurate summary of approach.</critic> <score>1</score>[GPT-5-mini: 1]

Step 181. The sequence x_n is defined recursively by $x_1 = a$ and $x_{n+1} = 2x_n + 1$. This sequence can be expressed explicitly as: $x_n = (a+1) \cdot 2^{n-1} - 1$
<critic>Correct explicit formula for x_n.</critic> <score>1</score>[GPT-5-mini: 1]

Step 182. The sequence y_n is defined as $y_n = 2^{x_n} - 1$. We need to find the largest k such that $y_1, \ldots, y_k$ are all prime.
<critic>Correct definition of y_n.</critic> <score>1</score>[GPT-5-mini: 1]

Step 183. We compute the first few terms of x_n and y_n for various values of a:
<critic>Valid computational approach.</critic> <score>1</score>[GPT-5-mini: 1]

Step 184. - For $a = 2$: - $x_1 = 2$, $y_1 = 2^2 - 1 = 3$ (prime) - $x_2 = 5$, $y_2 = 2^5 - 1 = 31$ (prime) - $x_3 = 11$, $y_3 = 2^{11} - 1 = 2047$ (composite) - Thus, $k = 2$.
<critic>Correct computations for a = 2.</critic> <score>1</score>[GPT-5-mini: 1]

Step 185. - For $a = 3$: - $x_1 = 3$, $y_1 = 2^3 - 1 = 7$ (prime) - $x_2 = 7$, $y_2 = 2^7 - 1 = 127$ (prime) - $x_3 = 15$, $y_3 = 2^{15} - 1 = 32767$ (composite) - Thus, $k = 2$.
<critic>Correct computations for a = 3.</critic> <score>1</score>[GPT-5-mini: 1]

Step 186. - For other values of a (such as 1, 4, 5, 6, 7, 15, etc.), we find that y_1 or y_2 is composite, making k less than 2.
<critic>Correct observation about other a values.</critic> <score>1</score>[GPT-5-mini: 1]

Step 187. After checking various values, we find that the maximum k is 2, achieved by $a = 2$ and $a = 3$.
<critic>Correct conclusion about maximum k.</critic> <score>1</score>[GPT-5-mini: 1]

Step 188. Thus, the largest possible k is: $\boxed{2}$
<critic>Correct final statement.</critic> <score>1</score>[GPT-5-mini: 1]

Table 14: **Qualitative example (OlympiadBench, prime-y_n sequence).** STC catches two independent errors in the thinking trace: step 75 makes a wrong claim about a=1 being the only a with y_1 prime; step 77 writes an incorrect expression for y_2. The final answer matches the ground truth.

Stepwise Think-Critique (Compact Mode) – AIME24 example

<|User|>: Jen enters a lottery by picking 4 distinct numbers from $S = \{1, 2, 3, \cdots, 9, 10\}$. 4 numbers are randomly chosen from S. She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers. The probability of her winning the grand prize given that she won a prize i...

Let's think step by step and output the final answer within \\boxed{}.

<|Assistant|><think>

Step 1. Okay, so I've got this probability problem here, and I need to figure it out step by step. Let me try to understand what's being asked.

Step 2. First, Jen is entering a lottery by picking 4 distinct numbers from the set S, which is 1,2,3,...,10. So, S has 10 numbers. Then, 4 numbers are randomly chosen from S. Jen wins a prize if at least two of her numbers are among those 4 chosen. If all four of her numbers are the chosen ones, she wins the grand prize. I need to find the probability that she wins the grand priz...

(. . . skipping 57 reasoning steps . . .)

Step 60. **Final Answer** $\boxed{116}$ </think>

(. . . skipping 7 reasoning steps . . .)

Step 68. Thus, the probability is $\frac{1}{115}$, and the sum of m and n where the probability is $\frac{m}{n}$ is $1 + 115 = 116$.

Step 69. $\boxed{116}$

Table 15: **Compact-mode output (AIME24, lottery probability).** The same problem as Table 11 but with the critique-instruction stripped from the system prompt. STC outputs reasoning only; the final answer matches the GT.

Stepwise Think-Critique (Compact Mode) – Minerva example

<|User|>: Preamble: A radio interferometer, operating at a wavelength of $1\ \mathrm{cm}$, consists of 100 small dishes, each $1\ \mathrm{m}$ in diameter, distributed randomly within a $1\ \mathrm{km}$ diameter circle. What is the angular resolution of a single dish, in radians?

Let's think step by step and output the final answer within \\boxed{}.

<|Assistant|><think>

Step 1. Okay, so I need to figure out the angular resolution of a single dish. Hmm, angular resolution... I think that's related to how well the telescope can resolve different sources of light, right? It's like the smallest angle at which two points can be distinguished. I remember something about the diffraction limit, maybe? So, the formula for the diffraction limit of a single s...

Step 2. Wait, the formula for the diffraction minimum in single-slit diffraction is given by $a \sin\theta = \lambda$, where a is the width of the slit or the aperture, and λ is the wavelength of the light. So, the angular resolution θ would be $\theta = \frac{\lambda}{a}$. Is that right? Let me confirm.

(. . . skipping 10 reasoning steps . . .)

Step 13. **Final Answer** The angular resolution of a single dish is $\boxed{0.01}$ radians. </think>

(. . . skipping 5 reasoning steps . . .)

Step 19. $\theta = \frac{0.01\,\mathrm{m}}{1\,\mathrm{m}} = 0.01$ radians

Step 20. Thus, the angular resolution of a single dish is $\boxed{0.01}$ radians.

Table 16: **Compact-mode output (Minerva, single-dish angular resolution).** Counterpart of Table 12.

Stepwise Think-Critique (Compact Mode) – MATH-500 example

<|User|>: The Greek army contained two types of soldiers: the upper class and the lower class soldiers. If there were a total of 5 upper class soldiers, and 10 lower class soldiers in a certain part of Athens, and the battle of Thermopylae demands a force of 4 upper class soldiers and 8 lower class soldiers, how many different battalions can be sent?

Let's think step by step and output the final answer within \\boxed{}.

<|Assistant|><think>

Step 1. Alright, so I have this problem about the Greek army and different types of soldiers. Let me try to figure it out step by step. Maybe I'll write down my thoughts as I go along.

Step 2. First, the problem says that the Greek army has two types of soldiers: upper class and lower class. There are 5 upper class soldiers and 10 lower class soldiers in a certain part of Athens. Then, the battle of Thermopylae demands a force of 4 upper class soldiers and 8 lower class soldiers. The question is asking how many different battalions can be sent.

(. . . skipping 29 reasoning steps . . .)

Step 32. **Final Answer** The number of different battalions that can be sent is $\boxed{225}$. </think>

(. . . skipping 4 reasoning steps . . .)

Step 37. Thus, the number of different battalions that can be sent is $\boxed{225}$.

Step 38. $\boxed{225}$

Table 17: **Compact-mode output (MATH-500, Greek army combinations).** Counterpart of Table 13.

Stepwise Think-Critique (Compact Mode) – OlympiadBench example

<|User|>: For a positive integer a, define a sequence of integers $x_1, x_2, \ldots$ by letting $x_1 = a$ and $x_{n+1} = 2x_n + 1$ for $n \geq 1$. Let $y_n = 2^{x_n} - 1$. Determine the largest possible k such that, for some positive integer a, the numbers $y_1, \ldots, y_k$ are all prime.

Let's think step by step and output the final answer within \\boxed{}.

<|Assistant|><think>

Step 1. Okay, so I have this problem here where I need to find the largest possible k such that for some positive integer a, the numbers y?, y?, ..., y? are all prime. The definitions given are x? = a, and each subsequent term is x_n+1 = 2x? + 1. Then y? is defined as 2^x? - 1.

Step 2. Hmm, let me try to break this down step by step. First, let's understand the sequence x?. It starts with some positive integer a, and each term is twice the previous term plus one. So, it's a recurrence relation.

(. . . skipping 123 reasoning steps . . .)

Step 126. **Final Answer** The largest possible k is $\boxed{2}$. </think>

(. . . skipping 7 reasoning steps . . .)

Step 134. Thus, the largest possible k such that $y_1, y_2, \ldots, y_k$ are all prime is $k = 2$.

Step 135. $\boxed{2}$

Table 18: **Compact-mode output (OlympiadBench, prime-y_n sequence).** Counterpart of Table 14.

Stepwise Think-Critique (Full Mode) – Minerva wrong-answer example (critic under-flags)

<|User|>: Subproblem 0: A particular star has an absolute magnitude $M = -7$. If this star is observed in a galaxy that is at a distance of $3\mathrm{Mpc}$, what will its apparent magnitude be? Solution: Given: $M = -7$ and $d = 3\mathrm{Mpc}$...

Let's think step by step and output the final answer within \\boxed{}.
Leverage the output \n\n as the indicator for separating reasoning steps. Following each reasoning step, please provide a critique and a score, where the critique reasons about the correctness of this step within <critic> and </critic> tags, the score of 1 for denoting correct or 0 denoting incorrect within <score> and </score> tags. Follow the format in this example precisely: [Your thinking process for a step goes here] <critic> [Your brief justification for the score goes here] </critic> <score> [0 or 1] </score>

<|Assistant|><think>

Step 1. Alright, so I have two subproblems here about magnitudes and distances. Let me tackle them one by one.
<critic>Clear organization of tasks; no mathematical errors.</critic> <score>1</score>[GPT-5-mini: 1]

Step 2. Starting with Subproblem 0: The star has an absolute magnitude of M = -7, and it's observed in a galaxy that's 3 Mpc away. I need to find its apparent magnitude. Hmm, okay. I remember that apparent magnitude is how bright the star appears from Earth, while absolute magnitude i...
<critic>Correct statement of the formula and units.</critic> <score>1</score>[GPT-5-mini: 1]

Step 3. So, let me plug in the numbers. The distance is given as 3 Mpc. Wait, Mpc stands for megaparsecs, right? So, 3 Mpc is 3,000 parsecs. Got it. So, d is 3,000 parsecs.
<critic>Correct conversion: 3 Mpc equals 3000 parsecs.</critic> <score>1</score>[GPT-5-mini: 0]

Step 4. Apparent magnitude m = -7 + 5 * log(d/10). Let me compute d/10 first. That's 3,000 / 10 = 300. So, log(300). Hmm, I need to remember what log base 10 of 300 is.
<critic>Correct substitution and computation of d/10.</critic> <score>1</score>[GPT-5-mini: 0]

Step 5. I know that log(100) is 2, and log(300) is log(3*100) = log(3) + log(100). Log(3) is approximately 0.4771, so log(300) $\approx$ 0.4771 + 2 = 2.4771.
<critic>Correct use of log properties and value for log 3.</critic> <score>1</score>[GPT-5-mini: 1]

(. . . skipping 49 thinking steps . . .)

Step 55. Therefore, I think the correct answers are approximately 5.39 for Subproblem 0 and 12.39 for Subproblem 1.
<critic>Correct restatement of final answers.</critic> <score>1</score>[GPT-5-mini: 0]

Step 56. **Final Answer**
<critic>Section header; no mathematical claim.</critic> <score>1</score>[GPT-5-mini: 1]

Step 57. The final answer for Subproblem 0 is $\boxed{5.39}$ and for Subproblem 1 is $\boxed{12.39}$.
</think>
<critic>Correct final numeric answers.</critic> <score>1</score>[GPT-5-mini: 0]

Step 58. Given the absolute magnitude $M = -7$ and the distance $d = 3$ Mpc:
<critic>Accurate restatement of given values.</critic> <score>1</score>[GPT-5-mini: 1]

Step 59. 1. **Apparent Magnitude Calculation:** - Convert the distance from Mpc to parsecs: $3\ \mathrm{Mpc} = 3000\ \mathrm{pc}$. - Use the formula for apparent magnitude: $m = M + 5\log_{10}\left(\frac{d}{10\ \mathrm{pc}}\right)$ - Substitute the values:...
<critic>Correct, detailed derivation and computation yielding 5.39.</critic> <score>1</score>[GPT-5-mini: 0]

(. . . skipping 1 summary blocks . . .)

Step 61. ### Final Answer
<critic>Section header; no mathematical content.</critic> <score>1</score>[GPT-5-mini: 1]

Step 62. The final answer for Subproblem 0 is $\boxed{5.39}$ and for Subproblem 1 is $\boxed{12.39}$.
<critic>Correct final answers restated.</critic> <score>1</score>[GPT-5-mini: 0]

Table 19: **Failure example (Minerva, apparent-magnitude formula)** – *critic under-flags*. At step 3 the model claims "3 Mpc equals 3000 parsecs" (the correct value is 3×10^6 parsecs); STC's critic confirms this with <score>1</score>("Correct conversion"), establishing a wrong distance that propagates through the entire trajectory. The pseudo-GT GPT-5-mini judge instead flags this with 0, reasoning: "*Incorrect unit conversion:* $1\,\mathrm{Mpc} = 10^6\,\mathrm{pc}$*, so* $3\,\mathrm{Mpc} = 3{,}000{,}000\,\mathrm{pc}$*, not* $3{,}000\,\mathrm{pc}$." GPT-5-mini further flags steps 4 onward as incorrect because the model continues to apply the wrong distance ("*Arithmetic using the incorrect distance is based on the wrong conversion*"), and the final boxed value as inconsistent with the correct distance modulus ≈ 27.39. The model arrives at the wrong final answer 12.39 (ground truth 27.39).