RAZORREWARD-RL: CAPABILITY-AWARE EFFICIENCY OPTIMIZATION FOR LLM REASONING VIA RAZOR REWARD

Anonymous authors
Paper under double-blind review

ABSTRACT

Chain-of-Thought (CoT) prompting is a de facto standard for unlocking strong reasoning performance in large language models (LLMs), but its step-by-step generation often introduces redundant computation in cost- and latency-sensitive production deployments. Existing reinforcement learning (RL) approaches for CoT efficiency rely on length-based rewards to mitigate redundant steps, but overlook the model's intrinsic query-specific reasoning capacity. This leads to two key limitations: for *Beyond-capability* queries that exceed the model's reasoning capacity, the reward mechanism encourages longer but futile reasoning, wasting massive token budgets; for *Near-capability* queries lying at the edge of the model's capacity, smoothed length-based rewards fail to provide clear discriminative signals to identify optimal stopping points, causing the model to either stop early with insufficient reasoning or drift away from the correct answer after redundant reasoning. To address these issues, we propose RazorReward-RL, a capability-aware RL framework for efficient CoT reasoning. As its name suggests, the core RazorReward function acts as a razor for reasoning chains. By segmenting CoT trajectories at natural reasoning step boundaries to construct semantically coherent prefixes, it sharply amplifies the reward gap in favor of the shortest sufficient prefix (the shortest prefix that yields a correct answer), imposes steep penalties for under- and over-reasoning on all *within-capability* and *near-capability* queries, while imposing heavy penalties for generation on *beyond-capability* queries to block futile reasoning. Experiments across multiple domains and model scales show that RazorReward-RL significantly reduces token consumption while improving accuracy, achieving a better accuracy–efficiency trade-off than prior methods.

1 INTRODUCTION

Chain-of-Thought (CoT) prompting (Wei et al., 2022) has enabled reasoning-oriented large language models (LLMs), such as DeepSeek-R1 (DeepSeek-AI et al., 2025) and QwQ (Team, 2025), to achieve strong performance across a wide range of benchmarks (Liu et al., 2025; Li et al., 2025; Chen et al., 2024b). However, in practical scenarios that are sensitive to inference cost and latency, such as search and question answering (Team et al., 2025), reasoning models often generate overly long, redundant, or even ineffective reasoning processes, leading to substantial token consumption and response delays. This phenomenon is commonly referred to as *overthinking* (Chen et al., 2024a).

To mitigate *overthinking*, recent reinforcement learning methods typically introduce length-based rewards to optimize the model's reasoning trajectory (Shao et al., 2024; Bai et al., 2022; Ouyang et al., 2022; Ramesh et al., 2024; Team et al., 2025; Hou et al., 2025; Yi & Wang, 2025). For example, S-GRPO (Dai et al., 2025) constructs prefixes of different lengths by truncating CoT trajectories, appends an early-stopping prompt, and then regenerates answers to construct positive and negative samples. Other methods, such as DAST (Shen et al., 2025) and ShorterBetter (Yi & Wang, 2025), sample multiple CoT trajectories and assign rewards based on both answer correctness and reasoning length. Although these methods enhance reasoning efficiency, their reward designs fail to account for the model's intrinsic reasoning capacity, which results in suboptimal performances.

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

To unpack this issue, we follow the truncation-based sampling framework proposed in prior work (Dai et al., 2025). Specifically, we split complete CoT trajectories into multiple contiguous prefixes at natural reasoning step boundaries, then evaluate whether each prefix can support the model in generating the correct final answer when an early-stopping prompt is appended. Based on the prefix correctness patterns exhibited by the target model, queries are categorized into three types aligned with the model's intrinsic reasoning capacity. For *Within-capability* queries, most or all prefixes produce correct answers, indicating that full-length CoTs are largely redundant for these problems. For *Near-capability* queries, different prefixes yield a mixture of correct and incorrect answers, suggesting that the model must reach a sufficient prefix before it can derive the right answer. For *Beyond-capability* queries, none of the prefixes can generate the correct answer, indicating that the query may fall outside the current reasoning capacity of the model.

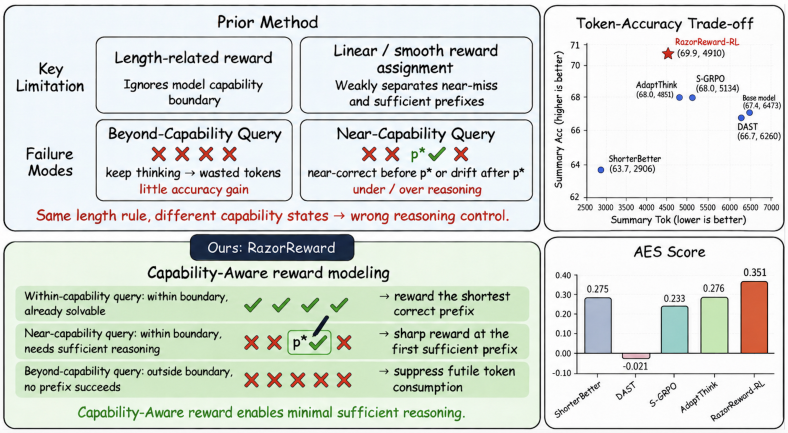

Figure 1: The mismatch between length-based rewards and the model's reasoning capacity in prior methods leads to two key limitations.

The mismatch between length-based rewards and the model's reasoning capacity in prior methods leads to two key limitations, as illustrated in Figure 1. First, for *Beyond-capability* queries, prior methods encourage longer reasoning in pursuit of a correct answer, which is inherently unachievable for this query type. This brings little accuracy gain, but causes massive token consumption. Second, for *Near-capability* queries, the core challenge is to locate the shortest sufficient prefix (i.e., the shortest prefix that can lead to a correct answer). Prior methods typically adopt a smoothed reward function that assigns similar scores to near-sufficient and sufficient prefixes, providing weak signals to separate them. This gives rise to two failure modes: the model may stop prematurely, leading to under-reasoning and incorrect outputs; conversely, it may continue generating steps after reaching a correct intermediate state, drifting to an incorrect final result.

To address the above limitations, we propose RazorReward-RL, a capability-aware efficiency optimization framework for LLM Reasoning. As implied by its name, the RazorReward function operates like a razor for reasoning chains. For *Within-capability* and *Near-capability* queries, RazorReward aims to make a precise cut at the first sufficient prefix; it assigns a high reward to this prefix, while severely penalizing prefixes that are either insufficient to derive the correct answer or include redundant reasoning steps after the sufficient point, thereby mitigating both under-reasoning and over-reasoning around the shortest sufficient prefix. For *Beyond-capability* queries, RazorReward suppresses futile lengthy reasoning to reduce token consumption. To operationalize this reward design, we segment reasoning trajectories at structural reasoning boundaries to construct semantically coherent prefixes. Across multiple benchmarks covering mathematics, science, and commonsense reasoning, as well as across models of varying scales (1.5B to 14B parameters), RazorReward-RL consistently achieves a better accuracy-efficiency trade-off than prior methods.

2 Related Work

Training-Free Methods These methods mitigate the overthinking problem without fine-tuning LLMs. For instance, Deer (Yang et al., 2025) dynamically halts generation by analyzing next-token

logits; Concise (Qiao et al., 2025) makes stopping decisions based on the model's confidence in intermediate answers. Differently, RouteLLM (Ong et al., 2025) employs routing strategies to assign queries to either strong or weak models, while ThinkSwitch (Liang et al., 2025) uses a lightweight regressor to switch between reasoning modes.

Supervised Fine-Tuning Methods LS-Mixture (Yu et al., 2025a) constructs a dataset that combines long CoT examples with structurally compressed short CoT examples for model fine-tuning. Similarly, Z1 (Yu et al., 2025b) creates a dataset containing both short and long CoT variants while removing explicit thinking markers. While these supervised fine-tuning (SFT) methods enable models to autonomously select reasoning modes, they place high demands on the quality of training data, requiring substantial manual effort.

Reinforcement Learning Methods AdaptThink (Zhang et al., 2025) dynamically selects between long and short reasoning modes, while Adacot (Lou et al., 2025) penalizes incorrect mode choices to prevent over/under-thinking. De-GRPO (Fang et al., 2025) prioritizes concise and correct reasoning through targeted rewards for efficiency-accuracy tradeoffs. Other methods adopt explicit length-based rewards: Training Efficient (Arora & Zanette, 2025) and LCPO (Aggarwal & Welleck, 2025) reward correct answers shorter than an average/preset length. DAST (Shen et al., 2025) adjusts reasoning length by question difficulty, encouraging longer chains for incorrect samples, whereas ShorterBetter (Yi & Wang, 2025) encourages lengths near first correct answer. S-GRPO (Dai et al., 2025) introduces serial sampling with early-stopped inference paths to construct sampling groups.

Our method differs from prior RL approaches in two key ways. First, unlike existing works that design reward functions in isolation from the model's reasoning capacity, we employ a razor-like capability-aware reward mechanism. Reasoning chains that either fall beyond the model's capacity or approach key steps but yield wrong answers incur strong penalties, forcing the model to identify essential steps within its competence, minimizing wasted effort on near-correct or unproductive reasoning. Second, instead of arbitrary segmentation to generate RL samples, we split trajectories at inherent structural reasoning boundaries to construct semantically coherent prefixes.

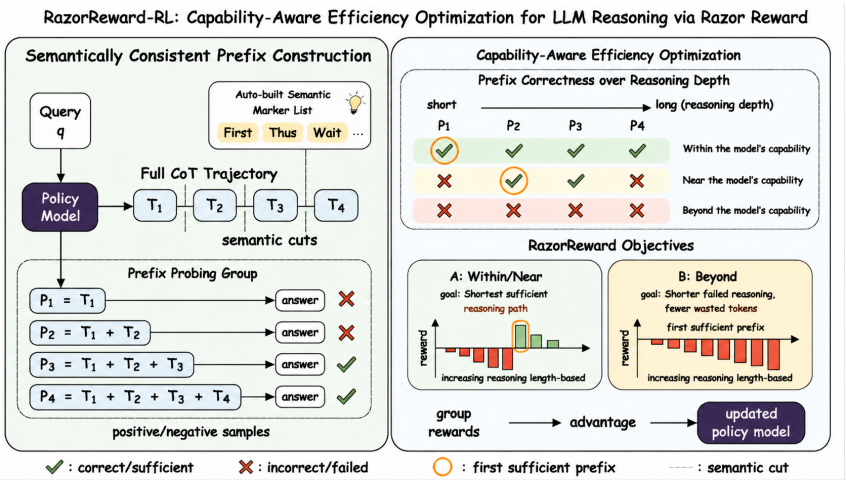

Figure 2: Overview of our method. The left part constructs semantically consistent prefixes from a full CoT trajectory. The right part probes prefix correctness over reasoning depth to identify capability regimes and applies the RazorReward function for RL optimization.

3 METHOD

In this section, we introduce RazorReward-RL, a capability-aware RL framework designed to boost LLM inference efficiency. As illustrated in Figure 2, the framework follows a two-stage pipeline: (1) construct semantically coherent reasoning prefixes from full CoT trajectories, (2) probe prefix correctness to classify the query relative to the model's intrinsic reasoning capacity, and optimize the policy model using the RazorReward function under a GRPO-style training objective.

3.1 SEMANTICALLY CONSISTENT PREFIX CONSTRUCTION

Given an input query q, the policy model π_θ first generates a complete chain of thought and final answer, i.e., $\langle s_0, c_0 \rangle = \pi_\theta(q)$, where $s_0 = [t_1, \ldots, t_n]$ is the token sequence representing the CoT, consisting of n reasoning tokens, and c_0 is the model's final answer derived from the CoT.

3.1.1 ADAPTIVE SEMANTIC MARKER LIST

A CoT typically consists of logically coherent reasoning blocks rather than arbitrary token spans. To eliminate dependence on pre-defined marker list, we employ an adaptive approach for constructing semantic markers tailored to the target model and dataset. Specifically, we sample policy-model reasoning trajectories on the training data, and use GPT-5 to identify recurring linguistic markers that indicate reasoning transitions, such as "wait", "thus", and "first". After deduplication and checks, this adaptive list—which updates with shifts in model or data—captures reasoning transitions, progressive steps, hypotheses, and summarizations, allowing robust prefix construction across styles and domains. Specific markers are detailed in Appendix B.

3.1.2 PREFIX PROBING GROUP

Based on the semantic markers defined in Section 3.1.1, we segment the CoT trajectory s_0 into $m+1$ continuous reasoning blocks $\{T_1, \ldots, T_{m+1}\}$ using the matched m segmentation points. For the i-th segmentation point, the prefix $s_i = T_1 \oplus \cdots \oplus T_i$ is generated. With the early stopping hint P_{es}, it is input to the policy model π_θ for solving:$c_i = \pi_\theta(q \oplus s_i \oplus P_{es})$. We aggregate all prefixes and their corresponding answers into a prefix probing group $G = \{\langle c_i, s_i \rangle\}_{i=1}^{m+1}$, ordered by reasoning depth, where $s_{m+1} := s_0$ denotes the complete trajectory appended after the deepest prefix. If c_i is correct, the sample is labeled as positive; otherwise, it is negative. Based on the distribution of positive examples in G, we determine the query q's position relative to the model's capability: fully correct queries are *Within-capability*, partially correct ones are *Near-capability*, and fully incorrect ones are *Beyond-capability*.

3.2 CAPABILITY-AWARE EFFICIENCY OPTIMIZATION

Using prefix group G, we optimize the policy via capability-aware rewards under a GRPO-style objective. The objective follows the core principle of GRPO, but differs in how the group is constructed and how rewards are defined: instead of sampling multiple complete responses, RazorReward-RL constructs an ordered prefix group along the same CoT trajectory, where prefixes are semantically continuous and sorted by reasoning depth. In this way, group-level comparison shifts from comparing multiple final responses to comparing states at different reasoning depths, allowing the samples to characterize under-reasoning, sufficiency, and post-sufficiency continuation, and enabling RazorReward to perform fine-grained reward assignment around the first sufficient prefix.

3.2.1 RAZORREWARD FUNCTION

The RazorReward function scores samples in G, extending beyond binary correctness to incorporate three components: correctness reward (i.e., R_{correct} prioritizes answer accuracy), format reward (i.e., R_{format} ensures structural compliance), and length reward (i.e., R_{length} promotes reasoning efficiency). We write $x_i := \langle s_i, c_i \rangle$ for the sample at index i of G. The total reward for sample $x_i \in G$ is: $R_{\text{RazorReward}}(x_i) = R_{\text{cor}}(x_i) + R_{\text{format}}(x_i) + R_{\text{len}}(x_i)$.

Correctness Reward: A sample receives 2 points if its answer is fully correct, otherwise 0. This prioritizes accuracy over efficiency or early termination, aligning with standard RL practices. **Format Reward**: To enforce reliable evaluation, a sample must adhere to the format `<think>...</think> <answer>...</answer>`—requiring all tags to appear exactly once in correct order. Compliance yields 1 point; non-compliance yields 0, eliminating ambiguity from inconsistent formatting. **Length Reward**: This component mitigates redundant reasoning via stepwise reward. The true optimal reasoning length is not directly observable: in principle, it would require knowing the shortest possible reasoning path that is sufficient for the model to answer correctly. We therefore use an operational approximation based on the prefix probing group. Let I_{right} denote the index of the first correct sample in G; this first correct prefix serves as the operational minimal sufficient prefix. If no prefix in G is correct, we set $\text{I}_{\text{right}} = m + 2$, treating the query as outside the current capability

under this trajectory and preferring shorter failed reasoning to avoid low-yield token spending. Note that the training objective depends only on $\mathrm{I}_{\text{right}}$ through the length reward: the within/near/beyond trichotomy is an interpretative framework over each group's prefix-correctness pattern rather than an explicit input to the objective, and the beyond-capability suppression emerges automatically when $\mathrm{I}_{\text{right}} = m + 2$. Let $\mathrm{N}^i_{\text{right}}$ denote the cumulative number of correct samples up to x_i. The length reward of x_i is defined as

$$R_{\text{len}}(x_i) = \begin{cases} \alpha^{\mathrm{N}^i_{\text{right}}-1}, & \text{if } c_i \text{ is } \checkmark. \\ -0.5 \cdot \alpha^{\mathrm{I}_{\text{right}}-i-1}, & \text{if } i < \mathrm{I}_{\text{right}} \text{ and } c_i \text{ is } \times. \\ -0.5, & \text{if } i > \mathrm{I}_{\text{right}} \text{ and } c_i \text{ is } \times. \end{cases} \tag{1}$$

The decay coefficient α controls the "razor-sharp" distinction strength of RazorReward around the first sufficient prefix. The interpretation is as follows: (1) **Correct samples**: The first correct prefix receives the highest reward, encouraging the model to stop reasoning once sufficiency is reached. The reward for subsequent correctness decays exponentially with α; (2) **Errors before the first correct prefix**: Penalties discourage under-reasoning, especially near the sufficient point where the model is close but not yet correct. The farther from the sufficient point, the penalty decays with α; (3) **Errors after the first correct prefix**: A fixed maximum penalty suppresses reasoning disrupted by further continuation after the correct prefix has appeared. Through this length reward, RazorReward jointly prioritizes the shortest sufficient CoT path for *Within* and *Near-capability* queries, discourages excessively long reasoning for *Beyond-capability* queries, and mitigates near-miss under-reasoning and post-sufficiency over-reasoning by applying sharp penalties around the first sufficient prefix in G.

3.2.2 GRPO-Style Training Objective

RazorReward-RL retains the group-relative optimization spirit of GRPO but shifts from sampling multiple complete responses to building an ordered prefix group along semantic boundaries within a single CoT, so advantage estimation compares the same problem at different reasoning depths; meanwhile, the RazorReward function drives the model to answer correctly as early as possible inside its reasoning capacity and to cut losses quickly outside it, aligning with cost-sensitive business needs.

For a query q, each sample $x_i \in G$ receives a scalar total reward $R_i := R_{\text{RazorReward}}(x_i)$. Following GRPO, its advantage is

$$\hat{A}_i = R_i - \frac{1}{m+1} \sum_{j=1}^{m+1} R_j, \tag{2}$$

where $m+1$ is the group size. For optimization, we represent every sample as an assembled response sequence. The complete sample is $o_{m+1} = s_{m+1} \oplus c_{m+1}$, while each early-stopped sample ($i \le m$) is $o_i = s_i \oplus P_{\text{es}} \oplus c_i$. The early-stopping instruction P_{es} is externally inserted context that conditions the generation of c_i; it is not an action sampled from the policy and is excluded from optimization.

We encode this distinction with a binary token-level mask $M_{i,t} \in \{0, 1\}$: $M_{i,t} = 1$ if $o_{i,t}$ belongs to s_i or c_i, and $M_{i,t} = 0$ if it belongs to P_{es}. Padding positions and positions after the terminal EOS are also zeroed in practice. The sample-level advantage is broadcast uniformly to policy-generated tokens, i.e., $\hat{A}_{i,t} = M_{i,t}\hat{A}_i$. The policy is then optimized using the clipped surrogate objective

$$\mathcal{J}(\theta) = \mathbb{E}_{q\sim P(Q),\, G\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{|G|} \sum_{i=1}^{m+1} \frac{1}{\sum_{t=1}^{|o_i|} M_{i,t}} \sum_{t=1}^{|o_i|} M_{i,t} \min\left(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\, \hat{A}_i \right) \right] \tag{3}$$

Here, q is an input query sampled from the training query distribution $P(Q)$, and the expectation is over the serial group G constructed from one complete rollout sampled from the reference policy. The token-level probability ratio is $\rho_{i,t} = \frac{\pi_\theta^{i,t}}{\pi_{\theta_{\text{old}}}^{i,t}}$, where π_θ and $\pi_{\theta_{\text{old}}}$ are the current and pre-update reference policies, respectively. We use $\pi^{i,t} = \pi(o_{i,t} \mid q, o_{i,<t})$ for the probability of token $o_{i,t}$ at position t in the assembled sequence. For an early-stopped sample, its preceding tokens include both the reasoning prefix s_i and P_{es}.

Table 1: Six-benchmark results across model scales. The best and second-best results per metric are marked in **bold** and underline, respectively. ‡S-GRPO results are from our reproduction.

Method	GSM8K		AIME24		AMC2023		MATH-500		GPQA_D		AIME25		Summary		
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	AES↑
DeepSeek-R1-Distill-Qwen-1.5B															
Vanilla	79.3	1,041	28.8	12,271	66.9	7,778	81.0	4,603	29.8	8,489	23.5	12,062	51.5	7,707	0.000
DEER	66.9	692	23.3	9,650	52.5	5,222	65.8	2,641	22.7	8,685	16.7	9,904	41.3	6,132	-0.787
ShorterBetter	71.6	**169**	19.6	**4,461**	70.0	**2,088**	72.2	**1,100**	28.8	**932**	15.6	**4,025**	46.3	**2,129**	0.216
AdaptThink	**83.3**	526	26.0	7,151	63.8	3,414	82.2	1,808	27.3	4,223	22.9	6,721	50.9	3,974	0.424
RazorReward-RL	82.7	677	**34.2**	9,115	**79.4**	4,551	**85.0**	2,710	**34.8**	5,511	**29.2**	7,209	**57.5**	4,962	**0.708**
DeepSeek-R1-Distill-Qwen-7B															
Vanilla	87.8	640	50.8	10,629	86.3	5,705	91.0	3,614	51.5	7,225	37.1	11,028	67.4	6,473	0.000
DEER	90.3	663	46.7	9,890	72.5	5,431	85.2	2,292	52.0	7,390	30.0	10,422	62.8	6,015	-0.273
ShorterBetter	84.7	**208**	46.7	**5,698**	82.5	**2,450**	86.0	**1,216**	51.0	**2,346**	31.2	**5,516**	63.7	**2,906**	0.275
AdaptThink	**91.1**	356	53.1	8,744	87.5	3,424	89.2	1,821	49.5	5,206	37.5	9,553	68.0	4,851	0.276
DAST	89.5	672	51.5	9,760	85.6	5,304	**93.0**	3,116	44.9	7,842	35.6	10,865	66.7	6,260	-0.021
S-GRPO‡	88.7	556	51.0	8,824	88.8	4,193	91.8	2,630	51.5	5,518	36.3	9,086	68.0	5,134	0.233
RazorReward-RL	90.5	612	**55.6**	7,905	**90.0**	4,117	92.4	2,529	**53.0**	5,623	**37.7**	8,675	**69.9**	4,910	**0.351**
DeepSeek-R1-Distill-Qwen-14B															
Vanilla	90.4	611	**63.1**	9,783	93.1	5,474	91.2	3,515	57.6	5,847	**44.2**	10,997	73.3	6,038	0.000
DEER	88.9	833	40.0	**8,261**	77.5	4,799	69.6	**2,588**	57.6	5,609	23.3	9,690	59.5	5,297	-0.818
RazorReward-RL	**92.0**	**607**	61.0	8,366	**95.0**	**4,161**	**92.8**	2,818	**61.6**	**4,609**	**44.2**	**9,409**	**74.4**	**4,995**	**0.221**

4 EXPERIMENT

4.1 SETUP

Datasets Following prior works (Dai et al., 2025), we evaluate our method on five math datasets: GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), AIME24 (MAA Committees, 2024), AIME25 (MAA Committees, 2025), AMC2023 (AI-MO, 2024) and science QA datasets: GPQA_D (Rein et al., 2023), which comprises 198 graduate-level questions.

For training, we also follow prior works to construct DeepMath-30K Balanced, a difficulty-balanced training set, by sampling problems from the DeepMath-103K dataset (He et al., 2025), which covers mathematics from grades 5 through 10. After pre-processing, we obtain a training set of 30190 samples. Dataset statistics and pre-processing details are provided in the Appendix C.

Baselines The RL-based methods we compare include ShorterBetter (Yi & Wang, 2025), DAST (Shen et al., 2025), and AdaptThink (Zhang et al., 2025). We also include S-GRPO (Dai et al., 2025), which likewise adopts serial truncation sampling, in our comparison. For training-free baselines, we include DEER (Yang et al., 2025). Additionally, we include Vanilla, the unmodified backbone model, which performs inference directly without any fine-tuning.

Implementation Details Key training hyperparameters—such as number of segmentation points, response length, batch size and learning rate—are detailed in the Appendix A. All models are evaluated with a 16K-token maximum response length. In line with prior work (Yi & Wang, 2025), we employ DeepSeek-R1-Distill-Qwen-7B and 1.5B as our primary backbones, and further report results on the 14B backbone to study scalability. All baseline models are faithfully reproduced using their officially released weights.

Evaluation Protocol We evaluate model performance using three primary metrics: **Accuracy (Acc↑)**, **Output Length (Tok↓)** and **Accuracy-Efficiency Score (AES↑)**. Output Length measures the average number of tokens generated per sample. The AES is introduced to jointly evaluate output brevity and accuracy preservation (Luo et al., 2025). A higher AES indicates better efficiency with minimal or no loss in correctness. The details of AES are provided in the Appendix D. Our evaluation protocol follows AdaptThink (Zhang et al., 2025), adopting the same evaluation functions and parameter settings.

4.2 Main Results

Six-benchmark results Table 1 reports the main results as equal-weight means over the six benchmarks. Relative to the matched Vanilla model, RazorReward-RL improves the overall accuracy by 6.0, 2.5, and 1.1 percentage points on the 1.5B, 7B, and 14B backbones, respectively, while reducing the average output length by 35.5%, 24.2%, and 17.3%. On the 1.5B backbone, RazorReward-RL improves accuracy on all six benchmarks, outperforming AdaptThink—the strongest baseline—by 6.6pp in accuracy, with its AES score (0.71) also leading the baselines by a large margin. On the 7B backbone, RazorReward-RL attains the highest Summary accuracy (69.9%) and AES (0.351) among all baselines, with particularly strong performance on MATH-500, GPQA, and AIME24. To demonstrate the generalization of our method to larger model scales, we further train on the 14B backbone, where RazorReward-RL maintains a positive accuracy gain while reducing the average output length by 17.3%, showing that the method scales with model size (no baseline releases 14B weights, so only the training-free DEER and Vanilla are compared here).

4.3 Ablation Study

This section reports the 7B application and ablation comparison (Table 2). Relative to Vanilla, RazorReward-RL improves the accuracy by 2.5pp and reduces token consumption by 24.2%. The contribution can be decomposed into two layers: replacing random segmentation with structured sampling mainly yields token-efficiency gains, reducing the Summary Tok from 5,134 to 4,792 (-342) while improving accuracy by 0.4pp; stacking the capability-aware Razor reward on top of structured sampling (S-GRPO with Structured Sampling $\rightarrow$ RazorReward-RL) mainly yields accuracy gains of +1.5pp (68.4 $\rightarrow$ 69.9), raising the AES from 0.303 to 0.351.

Table 2: The ablation results on six reasoning benchmarks (DeepSeek-R1-Distill-Qwen-7B backbone)

Method	GSM8K		AIME24		AMC2023		MATH-500		GPQA_D		AIME25		Summary		
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	AES↑
Vanilla	87.8	640	50.8	10,629	86.3	5,705	91.0	3,614	51.5	7,225	37.1	11,028	67.4	6,473	-
S-GRPO	88.7	556	51.0	8,824	88.8	4,193	91.8	2,630	51.5	5,518	36.3	9,086	68.0	5,134	0.233
S-GRPO with Structured Sampling	90.0	549	53.5	8,293	88.8	3,888	90.4	2,463	49.5	5,073	38.1	8,485	68.4	4,792	0.303
RazorReward-RL	90.5	612	55.6	7,905	90.0	4,117	92.4	2,529	53.0	5,623	37.7	8,675	69.9	4,910	0.351

4.4 Further Analysis

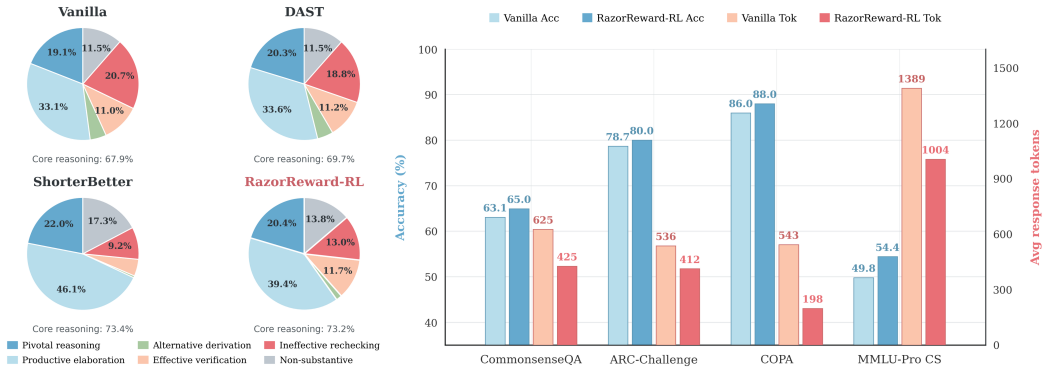

Figure 3: Left: Reasoning-trace behavior composition across different models (AMC2023). Right: Out-of-distribution (OOD) generalization performance: accuracy and token under 7B model.

Reasoning Trace Analysis We analyze AMC2023 traces with `qwen3.5-plus-2026-04-20`. The judge assigns each trace token proportion to six mutually exclusive behaviors: *Pivotal Reasoning*, *Productive Elaboration & Calculation*, *Productive Alternative Derivation*, *Effective Verification & Correction*, *Ineffective Rechecking / Repeated Exploration*, and *Non-substantive Statements*. As shown in Fig. 3 (Left), RazorReward-RL retains 73.2% of tokens in the four productive or answer-directed categories, compared with 67.9% for Vanilla, while reducing ineffective rechecking from

Table 3: Cross-lingual generalization results on Chinese math benchmarks. Best and second-best Acc and AES scores are **bold** and underlined, respectively.

Method	Gaokao-MathQA		OlymMATH-Easy		Summary		
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	AES↑
Vanilla	81.5	3862	16.0	13016	48.7	8439	–
DAST	80.9	4034	18.0	11738	49.5(+1.47%)	7886(-6.56%)	0.110
ShorterBetter	67.5	**1339**	13.0	**6789**	40.3(-17.40%)	**4064**(-51.84%)	-0.351
AdaptThink	72.9	2697	14.0	11365	43.5(-10.82%)	7031(-16.69%)	-0.374
RazorReward-RL	**85.8**	3048	**25.0**	12091	**55.4**(+13.61%)	7570(-10.30%)	**0.511**

Table 4: Performance comparison across sampling-defined MATH-500 capability regimes. E:C is mean erroneous-rollout tokens divided by mean correct-rollout tokens.

	Within-capability			Near-capability			Beyond-capability		
Model	Acc↑	Tok↓	E:C	Acc↑	Tok↓	E:C	Acc↑	Tok↓	E:C
Vanilla	97.7	1,540	2.74	72.4	2,912	2.01	0.0	8,901	–
RazorReward-RL	**99.7**	**1,351**	1.44	**85.2**	**2,473**	2.64	**5.9**	**6,130**	1.04
DAST	100.0	2,249	–	88.6	4,102	3.71	11.8	8,545	1.75
AdaptThink	97.7	1,138	5.22	82.4	2,527	2.79	5.9	6,822	6.17

20.7% to 13.0%. Its effective-verification share remains comparable to Vanilla (11.7% versus 11.0%), consistent with reducing repeated checking rather than indiscriminately removing verification.

We further evaluate reasoning sufficiency with `qwen3.5-plus-2026-04-20` on whole MATH-500 traces along four dimensions: overall logical quality, global coherence, reasoning usefulness, and verification quality. Training samples (the first truncated CoT reaching a correct answer during rollout) exhibit high logical completeness (avg. 0.95/1.00; each sub-metric 4.85/5). On inference outputs, RazorReward-RL attains the highest Overall Logical Quality (4.888) and Global Coherence (4.916) among five compared models, indicating that token reduction does not stem from premature guessing or broken logic. Detailed results are in Appendix E.1 and Table 8.

A further analysis of verification behavior (Appendix E.2 and Table 9) shows that RazorReward-RL matches Vanilla in overall verification rate (88.8%) while retaining a high rate among correct responses (95.1%) and reducing it among erroneous responses (31.3%).

Generalization of Adaptive Semantic Markers We further probe the cross-lingual generalization of the adaptive semantic marker construction (Section 3). We train RazorReward-RL on GSM8K-zh, derive a Chinese semantic marker list from sampled reasoning traces using the same pipeline, and evaluate on Gaokao-MathQA (Zhong et al., 2024) and OlymMATH-zh-easy (Sun et al., 2025).

As shown in Table 3, RazorReward-RL achieves a +13.61pp average accuracy gain, reduces token usage by 10.3%, and obtains a best AES of 0.51. This suggests that the segmentation mechanism benefits from adapting semantic markers to the target reasoning distribution, rather than relying on a fixed marker list tied to a specific language or backbone.

Performance under Sampling-Defined Capability Buckets Table 4 partitions MATH-500 into within-/near-/beyond-capability buckets by whether all, some (1–7), or none of Vanilla's 8 rollouts are correct. The within bucket is near saturation, and the main gains concentrate in the near bucket: RazorReward-RL raises accuracy from 72.4% to 85.2% (+12.8pp) while cutting the average output from 2,912 to 2,473 tokens (-439), the best token efficiency among all compared methods, whereas DAST attains higher accuracy (88.6%) at far greater token cost. In the beyond bucket, which mostly exceeds the models' reliable solving range, RazorReward-RL still produces the shortest responses (31.1% shorter than Vanilla).

Does Forced Continuation Recover Beyond-Capability Queries? A natural question is whether, for problems beyond the model's capability, further increasing the reasoning budget can still recover correct answers. We take the frozen beyond-capability subsets on which all 8 sampled responses fail,

Table 5: The performance of continued reasoning on beyond-capability questions (Vanilla 7B)

	MATH-500			AIME 2024+2025		
Stage	Acc↑	Cumulative Tok↓	Added Tok↓	Acc↑	Cumulative Tok↓	Added Tok↓
Concise (L_1)	0.0%	3,342	–	0.0%	4,096	–
1st continuation ($L_1 + L_2$)	2.9%	6,047	2,705	2.5%	7,819	3,723
2nd continuation ($L_1 + L_2 + L_3$)	0.0%	8,535	2,488	0.0%	9,952	2,134

and evaluate each chain successively at three cumulative stages: concise reasoning (L_1), the first forced continuation ($L_1 + L_2$), and the second forced continuation ($L_1 + L_2 + L_3$).

As shown in Table 5, additional continuation does not bring stable accuracy recovery: on both datasets, only one correct chain appears at the first continuation (MATH-500 0.0% $\rightarrow$ 2.9%, AIME 0.0% $\rightarrow$ 2.5%), and the second continuation returns to zero, while the cumulative tokens rise from 3,342 to 8,535 on MATH-500 (5,193 more in total, a surge of 155%) and from 4,096 to 9,952 on AIME (5,856 more in total, a surge of 142%). This indicates that the benefit of extra reasoning on beyond-capability problems is largely saturated: longer traces mainly consume more computation and bring only marginal (or even zero) accuracy improvement.

Table 6: Test-time scaling behavior on MMLU-Pro CS subsets. Pass-1 denotes the first-round answer, and Final denotes the result after a single continuation.

Subset	Model	Pass-1 Acc↑	Final Acc↑	Pass-1 Tok↓	New-Phase Tok↓	Final Cumulative Tok↓
Near capability (225)	Vanilla	56.9%	59.6%	2,642	**1,037**	3,679
Near capability (225)	RazorReward-RL	**60.9%**	**63.1%**	**2,204**	1,412	**3,616**
Beyond capability (72)	Vanilla	5.6%	5.6%	3,211	**1,640**	**4,851**
Beyond capability (72)	RazorReward-RL	5.6%	5.6%	**3,056**	2,045	5,100

Does RazorReward Preserve Useful Test-Time Scaling? Another question is whether penalizing long reasoning chains on beyond-capability queries could weaken the model's ability to benefit from test-time scaling on other query types (e.g., near-capability queries). We first emphasize the mechanism-level guarantee: as long as a trajectory reaches the correct answer, it receives a positive reward regardless of its length; RazorReward mainly penalizes long and ineffective trajectories beyond the model's capability, targeting low-yield overthinking rather than effective deep exploration.

To verify this, we partition MMLU-Pro-CS into near-capability and beyond-capability subsets. As shown in Table 6, the two layers diverge clearly: on the near layer, continuation benefits both models (Vanilla +2.7pp; RazorReward-RL +2.2pp), and RazorReward-RL leads at both stages (+4.0pp at Pass-1, +3.6pp at Final) with a first-round output 438 tokens shorter; on the beyond layer, both models stay at 5.6% with no gain from continuation. Thus, RazorReward preserves test-time scaling where continuation pays off, while avoiding returnless extra budget in the saturated beyond regime.

Generalizability in Out-of-Distribution (OOD) Scenarios As shown in Figure 3 (Right), RazorReward-RL improves accuracy over Vanilla 7B on all four non-math OOD benchmarks—CommonsenseQA (Talmor et al., 2019), ARC-Challenge (Clark et al., 2018), COPA (Wang et al., 2019), and MMLU-Pro Computer Science (Wang et al., 2024)—by 1.9, 1.3, 2.0, and 4.6 percentage points, respectively, while reducing mean response tokens on each dataset. The learned compact reasoning pattern thus transfers beyond the math-focused training distribution.

5 Conclusion

We presented RazorReward-RL, a capability-aware reward that sharply rewards the shortest sufficient prefix and suppresses futile generation beyond capability. Across three backbone scales and six benchmarks, it reduces token consumption by 17–36% while improving summary accuracy, with the largest gains in the near-capability regime; the learned policy further preserves useful test-time scaling and reallocates verification toward correct trajectories.

AI USE DISCLOSURE

We used generative AI tools for the following research tasks. First, we used an LLM judge (Qwen3.5-Plus-2026-04-20) for qualitative reasoning-trace analysis: categorizing trace segments into behavioral classes (Fig. 3, left), scoring reasoning quality along four dimensions (Table 8), and detecting verification steps (Table 9). Second, we used GPT-5 to surface candidate semantic-marker expressions from sampled reasoning trajectories during marker-list construction; all candidates were manually filtered and sanity-checked by the authors. Third, we used LLM assistance to polish parts of the paper's text.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: controlling how long A reasoning model thinks with reinforcement learning. *CoRR*, abs/2503.04697, 2025. doi: 10.48550/ARXIV.2503.04697. URL https://doi.org/10.48550/arXiv.2503.04697.

AI-MO. Amc 2023, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *CoRR*, abs/2502.04463, 2025. doi: 10.48550/ARXIV.2502.04463. URL https://doi.org/10.48550/arXiv.2502.04463.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of o1-like llms. *CoRR*, abs/2412.21187, 2024a. doi: 10.48550/ARXIV.2412.21187. URL https://doi.org/10.48550/arXiv.2412.21187.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=KuPixIqPiq.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Muzhi Dai, Chenxu Yang, and Qingyi Si. S-GRPO: early exit via reinforcement learning in reasoning models. *CoRR*, abs/2505.07686, 2025. doi: 10.48550/ARXIV.2505.07686. URL https://doi.org/10.48550/arXiv.2505.07686.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM learns when to think. *CoRR*, abs/2505.13379, 2025. doi: 10.48550/ARXIV.2505.13379. URL https://doi.org/10.48550/arXiv.2505.13379.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *CoRR*, abs/2504.11456, 2025. doi: 10.48550/ARXIV.2504.11456. URL https://doi.org/10.48550/arXiv.2504.11456.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *CoRR*, abs/2504.01296, 2025. doi: 10.48550/ARXIV.2504.01296. URL https://doi.org/10.48550/arXiv.2504.01296.

Xiaoyuan Li, Moxin Li, Rui Men, Yichang Zhang, Keqin Bao, Wenjie Wang, Fuli Feng, Dayiheng Liu, and Junyang Lin. Hellaswag-pro: A large-scale bilingual benchmark for evaluating the robustness of llms in commonsense reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 9038–9072. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.472/.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. Thinkswitcher: When to think hard, when to think fast. *CoRR*, abs/2505.14183, 2025. doi: 10.48550/ARXIV.2505.14183. URL https://doi.org/10.48550/arXiv.2505.14183.

Jiayu Liu, Zhenya Huang, Qi Liu, Zhiyuan Ma, Chengxiang Zhai, and Enhong Chen. Knowledge-centered dual-process reasoning for math word problems with large language models. *IEEE Trans. Knowl. Data Eng.*, 37(6):3457–3471, 2025. doi: 10.1109/TKDE.2025.3556367. URL https://doi.org/10.1109/TKDE.2025.3556367.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *CoRR*, abs/2505.11896, 2025. doi: 10.48550/ARXIV.2505.11896. URL https://doi.org/10.48550/arXiv.2505.11896.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *CoRR*, abs/2501.12570, 2025. doi: 10.48550/ARXIV.2501.12570. URL https://doi.org/10.48550/arXiv.2501.12570.

MAA Committees. American invitational mathematics examination (aime) aime 2024-i & ii. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024., 2024.

MAA Committees. American invitational mathematics examination (aime) 2025-i & ii. https://huggingface.co/datasets/opencompass/AIME2025, 2025.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. Concise: Confidence-guided compression in step-by-step efficient reasoning. *CoRR*, abs/2505.04881, 2025. doi: 10.48550/ARXIV.2505.04881. URL https://doi.org/10.48550/arXiv.2505.04881.

Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. Group robust preference optimization in reward-free RLHF. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=PRAsjrmXXK.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

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

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. DAST: difficulty-adaptive slow-thinking for large reasoning models. *CoRR*, abs/2503.04472, 2025. doi: 10.48550/ARXIV.2503.04472. URL https://doi.org/10.48550/arXiv.2503.04472.

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

Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, Lei Fang, and Ji-Rong Wen. Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models, 2025. URL https://arxiv.org/abs/2503.21380.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4149–4158, 2019. URL https://aclanthology.org/N19-1421.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan Yang. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025. doi: 10.48550/ARXIV.2501.12599. URL https://doi.org/10.48550/arXiv.2501.12599.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1905.00537.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.01574.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. Dynamic early exit in reasoning models. *CoRR*, abs/2504.15895, 2025. doi: 10.48550/ARXIV.2504.15895. URL https://doi.org/10.48550/arXiv.2504.15895.

Jingyang Yi and Jiazheng Wang. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *CoRR*, abs/2504.21370, 2025. doi: 10.48550/ARXIV.2504.21370. URL https://doi.org/10.48550/arXiv.2504.21370.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *CoRR*, abs/2505.03469, 2025a. doi: 10.48550/ARXIV.2505.03469. URL https://doi.org/10.48550/arXiv.2505.03469.

Zhaojian Yu, Yinghao Wu, Yilun Zhao, Arman Cohan, and Xiao-Ping Zhang. Z1: efficient test-time scaling with code. *CoRR*, abs/2504.00810, 2025b. doi: 10.48550/ARXIV.2504.00810. URL https://doi.org/10.48550/arXiv.2504.00810.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adaptthink: Reasoning models can learn when to think. *CoRR*, abs/2505.13417, 2025. doi: 10.48550/ARXIV.2505.13417. URL https://doi.org/10.48550/arXiv.2505.13417.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2299–2314, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.149. URL https://aclanthology.org/2024.findings-naacl.149/.

A Training Detail

Our method is implemented using the open-source RL framework VeRL (Sheng et al., 2024). Each query is segmented into at most 8 reasoning steps, and Rollout $= 8$ denotes eight complete rollouts sampled per query per update, each of which yields one serial prefix group. All models are evaluated with a 16K-token maximum response length; the continued-reasoning diagnostic in Table 5 uses a 4K-token cap for its concise stage.

Experiments use 8×A100 GPUs, with a single training session lasting approximately 8–12 hours. Models are initialized from DeepSeek-R1-Distill-Qwen (1.5B, 7B, 14B) checkpoints, with baselines reproduced from official releases.

Selected hyperparameters are summarized in Table 7.

Table 7: Key Training Hyperparameters

Hyperparameter	**Value**
Model variants	DeepSeek-R1-Distill-Qwen (1.5B, 7B, 14B)
Global batch size	128
Learning rate	1×10^{-6}
KL loss coefficient (β)	0.001
KL loss type	low-variance KL
Rollout	8
Tensor parallelism	4
GPU memory utilization	0.5–0.6
Decay weight α	1/3 or 1/2

A.1 Operational Definition of Capability Regimes

During training, each sampled query is assigned to a capability regime according to the prefix-correctness pattern of its probing group: if all prefixes answer correctly, the query is *within-capability*; if some (but not all) prefixes answer correctly, it is *near-capability*; and if no prefix answers correctly, it is treated as *temporarily beyond-capability*. At inference time, we operationalize the same definition with eight probes of the vanilla backbone model: queries with 8/8 correct probes are within-capability, those with 1–7/8 correct probes are near-capability, and those with 0/8 correct probes are beyond-capability. These labels describe the current behavior of the model under a fixed protocol rather than intrinsic properties of the questions.

B Enumeration of Reasoning Markers

As described in Section 3.1, our prefix construction relies on semantic markers that indicate natural transition points in a chain of thought. Rather than using a fixed hand-designed lexicon, we construct the marker list adaptively for the target model and training distribution. Specifically, we sample reasoning trajectories from the policy model on the training data and use the OpenAI API with GPT-5 to identify recurring linguistic expressions that mark reasoning transitions. After deduplication and sanity checking, we obtain the following marker list.

The resulting marker list used in our main English experiments covers five functional categories: transitions and self-verification, sequential progressions, step indicators, analytical or hypothetical constructs, and summarization expressions. These categories are used only to organize the markers for readability; during prefix construction, all markers are treated as candidate semantic boundary indicators.

1. **Transitions and Self-Verification**
   Markers that indicate a shift in reasoning, self-checking, reconsideration, or alternative exploration. Examples include: *but*, *wait*, *hold on*, *hmm*, *let's check*, *let me double-check*, *alternatively*.

2. **Sequential Progressions**
   Markers that indicate logical progression between reasoning steps. Examples include: *so*, *after that*, *next*, *then*, *therefore*, *given that*, *together*, *in total*, *thus*, *alright*, *finally*, *first*, *now*.

3. **Step Indicators**
   Markers that explicitly denote stepwise reasoning or numbered solution stages. Examples include: *step 1*, *step 2*, *1.*, *2.*, *to find*.

4. **Analytical and Hypothetical Constructs**
   Markers that introduce analysis, assumptions, hypotheses, or recalled information. Examples include: *let's analyze*, *let's consider*, *suppose*, *assume that*, *notice that*, *recall that*.

5. **Summarization and Conclusion**
   Markers that indicate summarization or the transition toward a conclusion. Examples include: *in summary*, *in conclusion*, *overall*.

English Marker List Used in Main Experiments

[["but", "wait", "hold on", "hmm", "let's check", "let me double-check", "alternatively"], ["so", "after that", "next", "then", "therefore", "given that", "together", "in total", "thus", "alright", "finally", "first", "now"], ["step 1", "step 2", "1.**", "2.**", "to find"], ["let's analyze", "let's consider", "suppose", "assume that", "notice that", "recall that"], ["in summary", "in conclusion", "overall"]]

For the Chinese cross-lingual experiment in Section 4, we follow the same adaptive construction pipeline on Chinese reasoning traces. Specifically, we sample Chinese CoT trajectories from the policy model on GSM8K-zh and use GPT-5 to identify recurring Chinese expressions that indicate semantic reasoning boundaries. After deduplication and sanity checking, we obtain the following Chinese marker list.

Chinese Semantic Marker List	
Transitions / Reflection 转折 / 反思	不过 但是 然而 但 可是 或者 反之 否则 另一种理解 另一种思路 另一种可能是 另一种方式
Progression / Causality 递进 / 顺承 / 因果	此外 另外 还有 同时 不仅如此 而且 还 更重要的是 进一步 并且 因此 所以 于是
Step Indicators 步骤标记	首先 第一 一开始 第一步 其次 接着 然后 接下来 之后 随后 现在 此时 同时 再次 那么 解得
Analysis / Assumptions 分析 / 假设	假设 设 令 如果 若 已知 根据 根据题意 根据题目 因为 由于 考虑 考虑到
Summary / Conclusion 总结 / 结论	综上 总之 综上所述 总的来说 结论是 由此可知 由此可得 最终 最后

Figure 4: Chinese Marker List Used in Cross-Lingual Experiments

C Training Dataset Construction Details

We construct the training set by stratified sampling from DeepMath-103K, a large-scale math problem dataset spanning grades 5-10, with difficulty levels annotated from 3 to 9 via multi-round AI assessment. To ensure training stability and representativeness, we first exclude samples with unclassifiable answers (non-numeric/yes-no), then compute the joint distribution of difficulty and topic. We sample 30,190 problems such that the marginal and conditional distributions over both difficulty and topic closely match those of the filtered source data.

To further improve training stability and prevent sudden increases in problem difficulty within batch, we reorder the sampled dataset so that the sum of difficulties in each batch (batch size 32 or 128) remains nearly constant across all batches. This is achieved via a greedy assignment ensuring that every batch's average difficulty is approximately 6.0, with minimal variance. Empirically, this leads to consistent learning dynamics and avoids training collapse due to abrupt difficulty spikes.

Sampling and balancing are implemented in Python, following a two-step process: (1) stratified sampling by difficulty and topic, (2) batch-wise reordering for difficulty balancing.

D Accuracy-Efficiency Score

The Accuracy-Efficiency (AE) Score, as proposed by (Luo et al., 2025), offers a metric for assessing whether a model can enhance inference efficiency—specifically, by shortening output length—while maintaining accuracy. The AE Score is formally defined by the following piecewise equation:

$$\text{AES} = \begin{cases} \Delta\text{Length} + \eta \cdot |\Delta\text{Acc}|, \text{if } \Delta\text{Acc} \geq 0 \\ \phi \cdot \Delta\text{Length} - \theta \cdot |\Delta\text{Acc}|, \text{if } \Delta\text{Acc} < 0 \end{cases} \tag{4}$$

where ΔLength denotes the percentage reduction in output length and ΔAcc the percentage change in accuracy, both relative to Vanilla. Following prior work (Luo et al., 2025), we set $\phi = 1$, $\eta = 3$, and $\theta = 5$. A higher AES indicates better efficiency with minimal or no loss in correctness.

E Supplementary Experiments and Analysis

E.1 Assessing Reasoning Sufficiency and Avoidance of Guessing

To further investigate whether the RazorReward mechanism induces insufficient reasoning or blind guessing, we conduct an in-depth evaluation of reasoning sufficiency on the MATH-500 dataset, using `qwen3.5-plus-2026-04-20` as the judgment model.

At the level of training samples, we analyze the first truncated chain of thought that can guess the correct answer (i.e., the first "correct-after-truncation" sample) obtained from rollout sampling. The results show that the overall logical completeness of these samples reaches an average of 0.95 out of 1.00, with step validity, coherence, and conclusion support each scoring around 4.85 out of 5. This indicates that the short-but-correct samples collected during training contain complete and sufficient reasoning, rather than coming from blind guessing.

At the level of inference outputs, we score the complete reasoning traces of the five models on the MATH-500 questions (one sampled response per model per question) along four dimensions: overall logical quality, global coherence, reasoning usefulness, and verification quality. The results are presented in Table 8.

Table 8: Whole-trace reasoning quality evaluation on MATH-500

Model	Overall Logical Quality	Global Coherence	Reasoning Usefulness	Verification Quality
RazorReward-RL	**4.888**	**4.916**	4.876	4.804
DAST	4.876	4.904	**4.884**	4.792
AdaptThink	4.876	4.908	4.880	4.500
Vanilla	4.868	4.852	4.852	**4.828**
ShorterBetter	4.420	4.636	4.432	3.884

As shown in Table 8, RazorReward-RL attains the highest Overall Logical Quality (4.888) and Global Coherence (4.916) among the five models; its Reasoning Usefulness (4.876) is essentially tied with DAST (4.884) and AdaptThink (4.880), and its Verification Quality (4.804) is second only to Vanilla (4.828). Together with the substantially shorter output length of RazorReward-RL in Table 2, these results show that its token reduction does not come from a degradation of logical quality: the trained model is no worse than any baseline in preserving logical structure and global coherence. Meanwhile, ShorterBetter ranks last on all four dimensions (4.420/4.636/4.432/3.884), showing that indiscriminate extreme compression does hurt reasoning quality, which in turn highlights that RazorReward's compression removes redundancy without breaking structure.

E.2 Verification Step Analysis

A potential concern is that verification steps are crucial for improving the accuracy of complex reasoning tasks. While excessive verification may be redundant, an appropriate amount of verification can enhance solution robustness. Here, we present a detailed analysis of how the RazorReward-RL reward mechanism interacts with verification behaviors.

The RazorReward-RL reward design does not indiscriminately penalize all verification steps. Instead, it aims to suppress redundant verification behaviors—specifically, repeated checking that does not improve correctness—while allowing for essential validation. This is achieved through the correctness-sensitive component of the reward function.

To empirically assess the effect on verification, we evaluate the five models on AMC2023 by sampling four generations per question across 40 questions, i.e., $n = 160$ responses per model, with `qwen3.5-plus-2026-04-20` as the verification judge. Table 9 reports accuracy, the overall verification rate, verification rates conditional on answer correctness, and mean output tokens (the AMC2023 Tok column of Table 1).

Table 9: Verification behavior and answer correctness on AMC2023.

Model	Acc.↑	Verification Rate	Correct Verification	Error Verification	Avg. Tokens
Vanilla	86.3%	88.8%	96.4% (138)	40.9% (22)	5,705
RazorReward-RL	**90.0%**	**88.8%**	95.1% (144)	**31.3% (16)**	**4,117**
DAST	85.6%	88.1%	95.6% (137)	43.5% (23)	5,304
ShorterBetter	82.5%	38.1%	43.9% (132)	10.7% (28)	2,450
AdaptThink	87.5%	61.9%	64.3% (140)	45.0% (20)	3,424

The results show that RazorReward-RL attains the highest accuracy (90.0%), while its overall verification rate (88.8%) is exactly the same as Vanilla's—indicating that the reward mechanism does not suppress verification behavior as a whole. The real difference lies in where verification is spent: among correct responses, RazorReward-RL maintains a high verification rate (95.1%, $n = 144$), whereas among erroneous responses the rate drops to 31.3% ($n = 16$), clearly below Vanilla (40.9%) and DAST (43.5%). In other words, RazorReward-RL verifies not less but more selectively: verification concentrates on correct trajectories while erroneous ones are checked far less often. Its mean output length (4,117 tokens) is also about 28% shorter than Vanilla's (5,705 tokens). Notably, ShorterBetter verifies far less often overall (38.1%) yet achieves only 82.5% accuracy, showing that simply doing less verification does not bring correctness—selective verification is the key.

E.2.1 Generalization to Other Domains and Models

We further examine the generalization of RazorReward-RL beyond the original mathematical reasoning setting. Figure 5 compares Vanilla 7B with RazorReward-RL 7B on four non-math benchmarks. RazorReward-RL improves accuracy on all four datasets—CommonsenseQA 63.1% → 65.0% (+1.9 pp), ARC-Challenge 78.7% → 80.0% (+1.3 pp), COPA 86.0% → 88.0% (+2.0 pp), and MMLU-Pro Computer Science 49.8% → 54.4% (+4.6 pp)—while reducing the mean response tokens on every dataset (e.g., from 625 to 425 on CommonsenseQA and from 1,389 to 1,004 on MMLU-Pro CS). This shows that the learned compact reasoning pattern transfers beyond the math-focused training distribution: the trained model is not only shorter out of domain, but also more accurate.

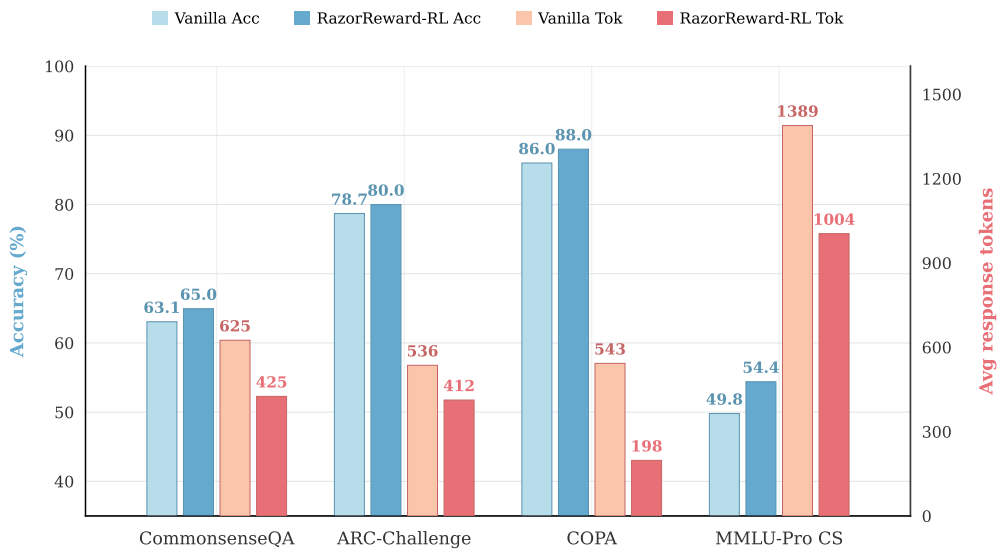

Figure 5: Out-of-distribution (OOD) generalization on four non-math benchmarks: accuracy and mean response tokens of Vanilla 7B versus RazorReward-RL 7B

E.3 Structural Analysis of Semantic Segmentation

Since RazorReward-RL constructs serial training samples through semantic segmentation, we further analyze how the structure of generated reasoning traces changes after training. This analysis helps us understand whether the model learns to produce more compact reasoning in a structurally meaningful way, rather than merely shortening outputs at the surface level.

We report training-log diagnostics for the 1.5B RazorReward-RL run. The recovered quantities are the cutpoint-aligned tokens and selected-prefix block lengths.

Training-stage segmentation statistics. We compare early and later windows from the completed 1.5B training log. Cutpoint and block statistics cover recorded selected prefix blocks and exclude the final tail block.

Table 10: Training-log segmentation diagnostics for the 1.5B RazorReward-RL run.

Snapshot	Prefix traces	Cutpoint Tok	Prefix Block Len.	Blocks $>$ 1024	Blocks $>$ 2048
Early	634	2,837 / 2,346	698 / 454	23.2%	6.4%
Later	623	1,745 / 1,217	439 / 255	10.2%	2.3%

As shown in Table 10, the logged later window has lower selected-prefix cutpoint tokens (2,837 to 1,745) and block length (698 to 439). The fractions of recorded prefix blocks exceeding 1,024 and 2,048 tokens decline from 23.2% to 10.2% and from 6.4% to 2.3%, respectively.

E.4 Truncation Distribution and Early-Exit Success Rate

RazorReward-RL relies on serial samples obtained by semantically truncating a complete reasoning trace. The quality of these truncation points directly affects the density of useful reward signals: if early-stopped prefixes were almost always incorrect, the resulting training groups would contain sparse positive samples and provide weak supervision. We therefore analyze both the distribution of truncation candidates and the success rate of early-exit inference at different truncation ranks.

As shown in Table 11, the parsed early-window rank accuracies range from 49.2% to 61.8%, and later-window values range from 73.0% to 75.9%. The smaller spread in the later window is descriptive of these logged serial groups.

Table 11: Serial-group rank accuracy from the 1.5B RazorReward-RL training log. Early and later values use parsed serial groups from the early and later training windows, respectively. Each rank denotes the k-th semantic truncation point within a logged serial family.

Truncation Rank	Early Acc	Later Acc
Rank 1	52.3%	75.2%
Rank 2	54.9%	75.2%
Rank 3	56.2%	75.4%
Rank 4	59.0%	74.9%
Rank 5	59.5%	75.1%
Rank 6	61.8%	75.5%
Rank 7	60.8%	75.9%
Rank 8	49.2%	73.0%

Table 12: Parsed serial-group statistics from the 1.5B RazorReward-RL training log.

Metric	Early	Later
Recorded groups	3,196	3,194
Positive samples per group (mean / median)	4.5 / 5	6.0 / 8
Groups with no positive sample	15.3%	10.6%
First correct at Rank 1	61.7%	84.1%
First correct within Ranks 1–2	80.3%	92.4%

The parsed group statistics in Table 12 show that, within the logged sample, positive labels become more common across the two windows: the mean rises from 4.5 to 6.0 per group, the median from 5 to 8, and the no-positive fraction falls from 15.3% to 10.6%. The logged first-correct-at-Rank-1 rate rises from 61.7% to 84.1%, while the within-Ranks-1–2 rate rises from 80.3% to 92.4%.

E.5 Further Analysis of the Ablation Study

The proposed reward function enhances model accuracy by incorporating a correctness component. This component explicitly guides the model to favor correct chains-of-thought (CoTs) over incorrect ones during training, thereby improving the accuracy of model outputs.

The significant improvement over S-GRPO primarily arises from two advantages of the semantic truncation strategy compared to random truncation:

- Constructing training samples via semantic truncation yields a higher proportion of positive samples compared to random truncation. Consequently, the model receives more positive reward signals during RL training. This was empirically validated: inference on the MATH-500 dataset using CoTs generated via semantic truncation showed a 2.75% higher accuracy than those using random truncation.
- The CoT paths generated using semantic truncation exhibit higher reasoning quality than those using random truncation. Specifically, using the qwen3.5-plus-2026-04-20 model as judge, we find that semantically truncated CoTs score higher on logical completeness and conclusion support.

These effects lead to improvements over S-GRPO.

E.6 Analysis of Redundant Reasoning Behaviors

Section 4.4 provides a qualitative and quantitative analysis of the reasoning process. To facilitate quantitative comparison, we follow the six-category partition of Appendix F and define four of them

Table 13: Sensitivity analysis of the decay coefficient α. The best and second-best results per metric are marked in **bold** and underline, respectively.

Setting	GSM8K		AIME24		AMC2023		MATH-500		GPQA_D		AIME25		Summary		
	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	Acc↑	Tok↓	AES↑
$\alpha = 1/2$	90.3	**514**	50.2	8,237	86.9	**3,805**	92.0	**2,163**	**56.6**	**4,610**	34.6	**8,273**	68.4	**4,600**	0.334
$\alpha = 1/3$	**90.5**	612	55.6	**7,905**	90.0	4,117	**92.4**	2,529	53.0	5,623	**37.7**	8,675	**69.9**	4,910	**0.351**
$\alpha = 2/3$	89.7	580	**57.5**	9,161	**90.6**	4,736	92.0	2,812	52.0	6,160	36.9	9,649	69.8	5,516	0.255

as **core reasoning**—Pivotal Reasoning, Productive Elaboration & Calculation, Productive Alternative Derivation, and Effective Verification & Correction—while the remaining two categories (Ineffective Rechecking / Repeated Exploration and Non-Substantive Statements) are the low-value content that the training process should compress as much as possible.

Quantitatively, on the complete reasoning traces of AMC2023, RazorReward-RL raises the share of core reasoning from 67.9% for Vanilla to 73.2% (+5.4 pp), while the combined share of low-value content (Ineffective Rechecking / Repeated Exploration + Non-Substantive Statements) falls from 32.1% to 26.8% (-5.4 pp). The reduction comes mainly from Ineffective Rechecking / Repeated Exploration (20.7% $\rightarrow$ 13.0%, -7.7 pp), while the share of Non-Substantive Statements rises slightly (11.5% $\rightarrow$ 13.8%); the gain in core reasoning comes mainly from Productive Elaboration & Calculation (33.1% $\rightarrow$ 39.4%, +6.4 pp) and Pivotal Reasoning (19.1% $\rightarrow$ 20.4%, +1.4 pp), with Effective Verification & Correction also rising slightly (11.0% $\rightarrow$ 11.7%), whereas the share of Productive Alternative Derivation drops (4.7% $\rightarrow$ 1.7%), indicating that the model focuses more on a single productive path than on multi-path exploration.

What distinguishes RazorReward-RL from the other models is not the absolute share of core reasoning—ShorterBetter (73.4%) has a similarly high share—but the composition of its core reasoning and its effective-vs-ineffective balance: its Effective Verification & Correction reaches 11.7%, the highest among the four models, and the gap between effective verification and ineffective rechecking is only 1.3 pp (11.7% vs. 13.0%), compared with 9.6 pp for Vanilla (11.0% vs. 20.7%) and 7.5 pp for DAST, indicating that the trained model spends almost every check where it matters. In contrast, ShorterBetter achieves its high core share through extreme compression, but has the lowest effective verification (4.8%) and the highest share of non-substantive statements (17.3%). This shows that the benefit of RazorReward-RL lies not in maximizing the share of core reasoning, but in reallocating tokens from repeated ineffective checking to effective verification and core reasoning, making the reasoning "short but well-grounded".

Manual inspection confirms this reallocation: after training, ineffective rechecking and repeated exploration (recomputing results, unsupported self-doubt, and cycling on false premises) are substantially reduced, while effective verification and correction of identified errors are retained.

RazorReward-RL demonstrates consistent improvements across tasks of varying difficulty:

- **GSM8K (within-capability):** RL training helps the model avoid redundant calculations and trivial mistakes.
- **AIME (beyond-capability):** The model decomposes structurally complex problems into logically rigorous steps and reduces erroneous explorations.

E.7 Sensitivity Analysis on Decay Coefficient α

We further evaluate the sensitivity of RazorReward-RL to the exponential decay coefficient α in the reward function, comparing three settings $\alpha = 1/2$, $\alpha = 1/3$, and $\alpha = 2/3$ on the 7B backbone (Table 13).

Overall, the three settings exhibit the expected trade-off: the $\alpha = 1/3$ setting attains the highest Summary Acc of 69.9% with 4,910 mean tokens and the best AES of 0.351, while $\alpha = 2/3$ reaches a nearly identical 69.8% at a higher token cost (5,516 tokens, AES 0.255), and $\alpha = 1/2$ minimizes token usage (4,600 tokens, AES 0.334) at a 1.5pp accuracy cost (68.4%). These results confirm that a sharper decay preserves accuracy at moderate token cost and yields the best overall trade-off, so we retain $\alpha = 1/3$ as the default decay coefficient in the main experiments.

E.8 Effect of the Early-Stop Prompt

In this subsection, we explore two questions: (1) do different early-stop prompts affect the model's responses, and (2) do different early-stop prompts affect the overall training process?

Two early-stop prompts have been used across our training runs. The neutral variant, used by the main run, is the short instruction "Time is limited, stop thinking and start answering."; the strict variant is the longer instruction "Stop reasoning now. Do not generate any further reasoning, analysis, explanation, or preamble. Output the final answer." Both are injected into the second-phase generation request after structured truncation and the two runs share the same configuration.

Table 14 summarizes the final six-benchmark results of the two runs. Regarding question (1), the strict-prompt run averages 3,284 response tokens over its recorded training steps, about 282 tokens (7.9%) below the neutral-prompt run's recorded mean of 3,566 tokens, i.e., the more explicit early-stop instruction is associated with shorter mean responses. Regarding question (2), the neutral-prompt run—our main model—attains an equal-weight Summary Acc of 69.9% with 4,910 mean tokens, while the strict-prompt run reaches 68.48% with 5,420 mean tokens. Overall, changes to the early-stop prompt still affect the final training results to some extent.

Table 14: Early-stop prompt variants and their final six-benchmark results.

Variant	GSM8K	AIME24	AMC23	MATH-500	GPQA	AIME25	Sum. Acc↑	Sum. Tok↓
Neutral-prompt run	90.5 / 612	55.6 / 7,905	90.0 / 4,117	92.4 / 2,529	52.97 / 5,623	37.7 / 8,675	**69.9%**	**4,910**
Strict-prompt run	88.40 / 565	49.58 / 9,311	91.88 / 4,345	91.40 / 2,912	55.05 / 5,792	34.58 / 9,593	68.48%	5,420

F Prompt Template for Reasoning Quality Analysis

Prompt for Reasoning Quality Analysis (Part 1)

You are a reasoning trace analyst. Your task is to classify the complete reasoning trace before the `</think>` tag by the actual role of each span in solving the problem. Partition the entire trace into mutually exclusive spans and assign exactly one of the six labels below to each span. Classify contribution rather than surface phrases such as "let me check" or "alternatively."

- Only analyze content before the `</think>` tag; return an empty list if the trace starts with that tag.
- Use exact, non-overlapping spans. A span may contain multiple sentences when they serve one function.

Categories:

- **Pivotal Reasoning:** An indispensable inference, constraint, or derivation needed for the final solution.
- **Productive Elaboration & Calculation:** Valid calculation, algebra, or detail that advances a viable solution.
- **Productive Alternative Derivation:** A distinct valid route, or an alternative that adds information and is used to advance the solution.
- **Effective Verification & Correction:** An independent check of an earlier result, or a correction that resolves an identified error, where the checked/corrected result is subsequently used or supports the final answer.
- **Ineffective Rechecking / Repeated Exploration:** Repeated computation, unsupported doubt, unresolved contradictions, cycling on a false premise, or guessed alternatives that add no usable constraint or conclusion.
- **Non-Substantive Statement:** Formatting, repeated conclusions, generic filler, or text unrelated to solving the problem.

Prompt for Reasoning Quality Analysis (Part 2)

Output Format:

- Return a JSON object with the following required keys: `behavior_proportions`, `effective_verification_events`, `ineffective_rechecking_events`, and `rationale`.
- `behavior_proportions` must contain exactly the six category names and numbers in $[0, 1]$ summing to 1 within 0.02.
- Each event list contains objects with `text` and `rationale`; use an empty list when the behavior is absent.

Additional Instructions:

- Stop processing as soon as the </think> tag is encountered. Do not classify anything beyond it.
- A check is effective only when it independently checks or corrects an earlier result and the checked/corrected result is used or supports the final answer.
- Do not label a step effective merely because the final answer is correct; use the reference answer only to assess whether a correction/check is mathematically valid.
- If a span could fit multiple categories, choose the most specific primary function.
- Do not infer correctness for unrelated reasoning spans. Use surrounding context to identify the reasoning flow.

Example Output: a JSON object containing the four required keys and the six category names.
Return only the JSON object, without any extra commentary or explanation.
Question: question
Reference answer: gold_answer
Complete response: response

LIMITATIONS AND IMPACT STATEMENT

Limitations. RazorReward-RL is primarily designed for cost- and latency-sensitive reasoning scenarios, with the goal of improving the accuracy–efficiency trade-off while reducing redundant reasoning. Therefore, the method is not intended to explore the upper bound of model reasoning capability under unlimited test-time compute. In settings where the primary objective is to maximize peak performance regardless of token cost, stronger test-time scaling or search-based methods may still be more suitable.

In addition, our categorization of within-, near-, and beyond-capability queries depends on the specific model, finite rollout budget, and current prompting setup. It should therefore be viewed as an operational definition rather than an absolute judgment of task solvability. Some queries labeled as beyond-capability may become solvable with stronger models, external tools, or larger sampling budgets.

Impact Statement. This work aims to reduce token consumption and response latency in reasoning models, thereby lowering deployment costs and improving usability under limited computational resources. It may also contribute to energy savings in large-scale inference. Meanwhile, shorter reasoning traces may reduce the visibility of intermediate reasoning steps for human auditing. Therefore, RazorReward-RL should be used together with task-specific safety evaluation, failure monitoring, and responsible deployment practices, rather than being regarded as a substitute for factuality, fairness, privacy, or safety alignment mechanisms.