APR: Penalizing Structural Redundancy in Large Reasoning Models via Anchor-based Process Rewards

Anonymous ACL submission

Abstract

Mitigating "overthinking" is crucial for the efficiency of Large Reasoning Models (LRMs). However, existing efficient reasoning methods typically rely on global length penalties, which indiscriminately suppress necessary exploratory reasoning steps and inevitably degrade task accuracy. In this paper, we introduce the *Reasoning Anchor* to identify the structural redundancy in LRMs: the *Answer-Stable Tail (AST)*, a post-anchor self-restatement phase following the model's first answer commitment. AST not only wastes computation but frequently causes the premature truncation of correct reasoning paths. Therefore, we propose *Anchor-based Process Reward (APR)*, a fine-grained reinforcement learning mechanism that surgically penalizes only the AST. This targeted supervision teaches LRMs to timely terminate post-anchor generation while keeping informative pre-anchor derivation intact. Extensive evaluations on 1.5B and 7B LRMs demonstrate an optimal performance-efficiency trade-off across mathematical and out-of-domain reasoning benchmarks, effectively proving that targeted redundancy reduction simultaneously optimizes task accuracy and generation length.

1 Introduction

In recent years, Large Reasoning Models (LRMs) (OpenAI, 2024; Team, 2024; Guo et al., 2025) have achieved significant performance breakthroughs on complex tasks through Test-Time Scaling (TTS) (Snell et al., 2024). However, this extension of reasoning length has triggered a widespread phenomenon known as *overthinking* (Chen et al., 2024; Luo et al., 2025a), where models generate substantial redundant behaviors that make no contribution to the final answer (Su et al., 2025). The resulting prohibitive computational costs heavily hinder the real-world scalability of LRMs.

To enhance reasoning efficiency, existing approaches within Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2024; Shao et al., 2024) paradigms primarily integrate global length penalties into the reward function, operating at either the response-level (Aggarwal and Welleck, 2025; Hou et al., 2025) or outcome-level (Luo et al., 2025a; Xiang et al., 2025). We argue that such coarse-grained constraints fundamentally suffer from the credit assignment problem: by indiscriminately penalizing the entire trajectory, they risk suppressing informative intermediate reasoning steps, subsequently impairing the model's overall reasoning performance.

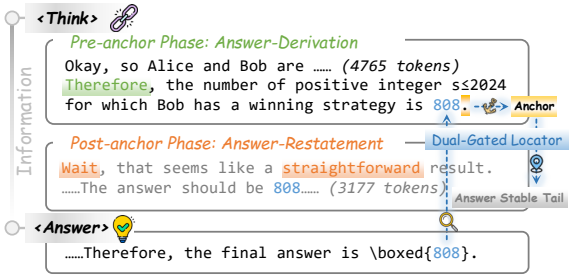

Figure 1: Schematic illustration of structural redundancy in LRMs. The reasoning anchor divides the trajectory into two phases: an information-dense pre-anchor phase dedicated to logical derivation and initial answer commitment, and an information-sparse post-anchor phase characterized by repetitive self-restatement, formally termed the *Answer-Stable Tail (AST)*.

In this paper, we reconceptualize overthinking from a fine-grained, information-gain perspective. As shown in Figure 1, we introduce the *Reasoning Anchor* to precisely identify the moment a model makes its first definitive commitment to a stable answer. Our extensive empirical analyses reveal an intrinsic structural redundancy across various LRM families: regardless of answer correctness, models typically engage in a meaningless, repetitive self-restatement immediately after the anchor without revising the initial conclusion. We formally term this information-sparse suffix the *Answer-Stable Tail (AST)*. Crucially, our preliminary study uncov-

ers that AST is not merely an efficiency bottleneck; it actively exhausts the token budget, causing otherwise correct reasoning paths to be prematurely truncated. Therefore, mitigating AST is a necessity to preserve reasoning accuracy.

Motivated by this insight, we propose *Anchor-based Process Reward (APR)*, a structure-aware reward shaping mechanism. Unlike prior global penalties, APR utilizes a highly reliable Dual-Gated Locator to identify the reasoning anchor and surgically penalizes only the redundant AST. This fine-grained supervision signal guides LRMs to compress the post-anchor answer-restatement while maintaining the pre-anchor answer-derivation, ultimately shortening reasoning trajectories while securing task performance.

Leveraging the DAPO policy optimization algorithm (Yu et al., 2025), we evaluate APR on 1.5B and 7B LRMs across five mathematical datasets and five out-of-domain (OOD) reasoning tasks. Benchmarked against five efficient-reasoning baselines, our APR models achieve a state-of-the-art balance between task accuracy and generation length. On mathematical benchmarks, our APR models effectively halve the reasoning generation length (a 45.8%–51.1% reduction) while yielding double-digit accuracy gains (13.7%–20.6%). Furthermore, despite being optimized exclusively on mathematical data, APR demonstrates robust generalization to diverse OOD tasks (e.g., coding and general logic). These findings validate that targeted redundancy reduction is the key to simultaneously optimizing reasoning performance and efficiency. The main contributions are three-fold:

- **New Perspective on Overthinking:** We define the Reasoning Anchor and identify the Answer-Stable Tail (AST) as a universal structural redundancy that unconditionally consumes token budgets and impairs accuracy.
- **Structure-Aware Reward Mechanism:** We propose Anchor-based Process Reward (APR), a fine-grained RL supervision signal that surgically penalizes the post-anchor AST, mitigating the accuracy degradation inherent in traditional global length constraints.
- **Superior Performance-Efficiency Trade-off:** Extensive experiments demonstrate that APR effectively rescues truncated reasoning paths, achieving an optimal accuracy-length balance and consistent out-of-domain generalization.

2 Related Work

2.1 Overthinking in LLM Reasoning

Test-Time Scaling (TTS) enhances the capabilities of Large Language Models (LLMs) by allocating more inference-time compute (Snell et al., 2024), primarily driven by the Reinforcement Learning with Verifiable Rewards (RLVR) paradigm (Shao et al., 2024; Lambert et al., 2024). However, this paradigm inadvertently decreases output entropy (Yue et al., 2025), causing models to generate excessively lengthy reasoning traces even for simple tasks—a phenomenon termed *overthinking* (Chen et al., 2024). While some works employ learned routers to delegate queries to smaller or larger models based on problem difficulty (Liang et al., 2025; He et al., 2025), these approaches rely on external modules and fail to address the intrinsic tendency of LRMs to generate redundant sequences.

2.2 Efficient Reasoning of LRMs

To intrinsically optimize reasoning efficiency, recent efforts integrate length penalties into the RL reward function. One line of work applies direct length constraints, such as group-wise length normalization (Team et al., 2025; Arora and Zanette, 2025) or fixed truncation thresholds (Aggarwal and Welleck, 2025). Another line proposes difficulty-adaptive penalties by coupling reasoning length with outcome accuracy (Xiang et al., 2025; Su and Cardie, 2025), for instance, by selecting the shortest correct rollout as the optimal target (Yi et al., 2025) or utilizing empirical accuracy to indicate difficulty (Liu et al., 2025).

However, these advancements operate fundamentally as coarse-grained constraints, failing to distinguish between valid logical exploration and actual verbosity. By indiscriminately penalizing the entire response, they suffer from the credit assignment problem, resulting in a sub-optimal trade-off between conciseness and task accuracy. In contrast, we identify the fundamental source of overthinking as the intrinsic structural redundancy within LRMs. Accordingly, our work shifts toward precise process supervision, exclusively penalizing the meaningless answer-restatements without harming effective derivations.

3 Preliminary Study: Structural Redundancy in Reasoning Traces

To achieve a fine-grained understanding of the "overthinking" phenomenon in Large Reasoning

Models (LRMs), we conduct a large-scale empirical study to investigate the microscopic behaviors of LRMs within their generation trajectories.

3.1 Reasoning Anchor and Answer-Stable Tail

Modern LRMs typically structure their responses into two distinct parts: a `<think>` block containing the intermediate reasoning process, followed by an `<answer>` block presenting the final conclusion. The well-known "overthinking" phenomenon predominantly occurs within this `<think>` block. Prior findings (Chen et al., 2024; Luo et al., 2025a) characterize redundancy restrictively as the sequence generated after the first occurrence of the *correct* answer. However, we observe that regardless of whether the final answer is correct or incorrect, the latter half of the reasoning trajectory frequently restates the model's derived conclusion multiple times without yielding any new information (see an illustrative case in Appendix A.2).

To formally quantify this phenomenon, we define the Reasoning Anchor, denoting the critical turning point where the model makes its first definitive commitment to a stable answer. Specifically, we segment the thinking process into a sequence of sentences $S = (s_1, s_2 \ldots, s_N)$, where N is the total number of sentences. Let $\mathrm{end}(s_i)$ denote the global token index of the last token of sentence s_i. Given a reference answer y_{ref} (typically the model's self-generated final answer, or the ground truth in truncated cases), we aim to locate the index $k^* \in \{1, \ldots, N\}$ of the specific sentence in which y_{ref} first explicitly emerges. The reasoning anchor is then strictly defined as the exact token position at the end of this sentence:

$$t_{\text{anc}}(S, y_{\text{ref}}) = \mathrm{end}(s_{k^*}) \tag{1}$$

This anchor naturally divides the reasoning trace into two phases. The *pre-anchor* phase involves effective derivation, trial-and-error, and the initial answer commitment. In contrast, the *post-anchor* phase consists almost entirely of redundant answer-restatements. Since the derived answer remains unmodified during this period, we formally define this information-sparse suffix spanning from the reasoning anchor to the end of the `<think>` block as the *Answer-Stable Tail (AST)*. The length of AST (L_{AST}) and its ratio (ρ) relative to the entire thinking length (L_{think}) are calculated as:

$$L_{\text{AST}}(S, y_{\text{ref}}) = L_{\text{think}} - t_{\text{anc}}(S, y_{\text{ref}}) \tag{2}$$

$$\rho = \frac{L_{\text{AST}}(S, y_{\text{ref}})}{L_{\text{think}}} \tag{3}$$

3.2 Dual-Gated Anchor Localization Method

Locating the reasoning anchor t_{anc} requires identifying the exact moment the reference answer first emerges within a lengthy reasoning trajectory. Simply applying string matching is vulnerable to false positives due to two inherent challenges: (1) The `think` block lacks standardized delimiters (e.g., `\boxed{}`), rendering regex-based extraction unreliable for capturing equivalent mathematical expressions. (2) The linguistic patterns denoting a conclusion are highly diverse. Common connectives (e.g., `so`) frequently precede intermediate steps rather than the actual final answer, thereby misleading simple keyword detectors.

Therefore, we propose a Dual-Gated Anchor Localization method tailored for discrete answers, which identifies the anchor sentence index k^* via a two-stage gating mechanism designed to verify both symbolic equivalence and contextual intent.

Semantic-Symbolic Gate. This gate first retrieves all candidate sentences that mathematically entail the reference answer. Let $\mathrm{Sym}(\cdot)$ be a function that extracts mathematical expressions using a formal algebra system (e.g., SymPy) and tests for symbolic equivalence against y_{ref}. The set of semantically matched sentences is:

$$\mathcal{I}_{\text{sym}} = \{i \in \{1, \ldots, N\} \mid \mathrm{Sym}(s_i) \equiv y_{\text{ref}}\} \tag{4}$$

Contextual Gate. To filter out noisy intermediate steps from the candidates, this gate requires explicit linguistic indicators. A valid anchor sentence must either contain a conclusion keyword or be immediately followed by a verification action. Let $\mathrm{Con}(\cdot)$ and $\mathrm{Ver}(\cdot)$ denote the presence of conclusion keywords and verification keywords, respectively (see Appendix A.1 for the detailed keyword lists). The set of contextually valid sentences is:

$$\mathcal{I}_{\text{ctx}} = \{i \in \{1, \ldots, N\} \mid \mathrm{Con}(s_i) \vee \mathrm{Ver}(s_{i+1})\} \tag{5}$$

The anchor sentence index k^* is strictly determined as the earliest sentence in the reasoning trajectory that satisfies both criteria:

$$k^* = \min\Big((\mathcal{I}_{\text{sym}} \cap \mathcal{I}_{\text{ctx}}) \cup \{N\}\Big) \tag{6}$$

If the intersection is empty, the default $k^* = N$ implies the anchor lies at the very end with zero AST redundancy.

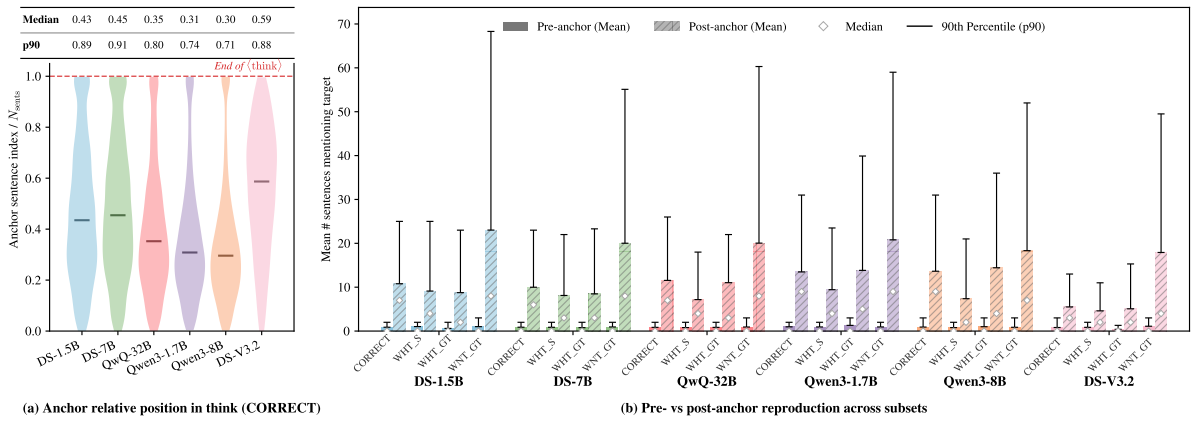

Figure 2: Comprehensive empirical analysis of AST across six representative LRMs. **(a)** Distribution of the reasoning anchor's relative position within the reasoning trace for correct responses. **(b)** The mean frequency of target answer occurrences in the pre-anchor versus post-anchor phases across four outcome subsets. Detailed statistical metrics for both subfigures are provided in Tables 4 and 5 of Appendix A.

3.3 Empirical Analysis of AST Redundancy

To investigate whether the Answer-Stable Tail (AST) is a prevalent phenomenon across different LRMs, we conducted a comprehensive statistical analysis spanning three LRMs (DeepSeek-R1-Distill-Qwen-1.5B/7B, QwQ-32B) as well as three LLMs in reasoning mode (Qwen3-1.7B/8B and DeepSeek-V3.2)[1]. We evaluated them on five mathematical benchmarks (AIME24, AMC, MATH500, Minerva, and OlympiadBench). With generation lengths truncated at $8,192$ tokens, we sampled 8 responses per query for all models (except for DS-V3.2), yielding a corpus of over 50K reasoning trajectories. To systematically capture model behaviors across various terminal states, we categorized the analyzable trajectories into four distinct subsets: `CORRECT`, `WRONG_HT_SELF` (incorrect but contains a stable self-generated answer), `WRONG_HT_GT` (incorrect, evaluated using the ground truth as a reference), and `WRONG_NT_GT` (truncated before the `<answer>` tag). Our key findings are as follows:

Observation 1: The Widespread Prevalence of AST. As illustrated by the violin plots in Figure 2(a), where an earlier emergence of the anchor corresponds to a larger AST proportion, it is evident that the AST phenomenon is not an isolated artifact but an intrinsic structure across various model scales and families. Detailed statistical metrics can be seen in Table 4. Specifically, the Qwen3 series exhibits the most severe redundancy, with the AST ratio peaking at 69% and the absolute AST length reaching 3,059 tokens. Similarly, the DeepSeek series demonstrates an average redundancy ratio of approximately 53%, corresponding to an AST length of 1,449 tokens. These quantitative results confirm the definitive existence of structural redundancy within LRMs.

Observation 2: Reliability of the Dual-Gated Locator. Table 5 reports the frequency of the reference answer appearing in sentences before and after the reasoning anchor. Notably, our dual-gated locator successfully identifies a valid anchor in over 99% of the `CORRECT` reasoning trajectories across all models. Furthermore, in the pre-anchor phase, the median frequency of target answer occurrences is strictly 0, with the mean remaining close to 1.0. This near-zero occurrence proves that our locator does not "miss" earlier definitive mentions of the answer, nor is it erroneously triggered by coincidental intermediate numbers. Together, the high localization coverage and the precise "first-commit" capturing robustly demonstrate that our proposed dual-gated locator is accurate and reliable, serving as a trustworthy foundation for quantifying the AST redundancy.

Observation 3: The Severity of Answer-Restatement. As shown in Figure 2(b) and quantified in Table 5, there is a pronounced contrast in the frequency of target answer occurrences before and after the reasoning anchor. Transitioning into the post-anchor phase, LRM behaviors undergo a fundamental shift from zero-occurrence derivation to high-frequency repetition. Furthermore, regardless of the reference answer utilized, the overall severity of this repetition remains con-

[1]For brevity, DeepSeek models are abbreviated as "DS".

sistent. Even when evaluated against the model's self-generated final answer, the conclusion is still restated extensively; specifically, the mean number of restatements ranges from 5 to 14 times, with the 90th percentile extending up to 31 times. These statistics confirm that AST is not a constructive self-correction or effective verification process, but rather an unconditional inertia of answer-restatement once an initial commitment is made.

Observation 4: The Critical Necessity to Mitigate AST. A closer inspection of the erroneous subsets (`WRONG_HT_GT` and `WRONG_NT_GT`) in Table 5 reveals that AST is not a harmless byproduct, but a structural flaw that actively impairs task accuracy. In the `WRONG_HT_GT` subset, we observe that models frequently alternate between candidate answers. Considered alongside the relative anchor positions across datasets in Table 4, the Qwen3 series particularly exhibits a strong tendency to be trapped in self-doubt on challenging tasks such as AIME24, ultimately deviating from correct derivations. Furthermore, in the prematurely truncated `WRONG_NT_GT` subset, the ground-truth is restated with the highest frequency among all subsets, averaging 18 to 23 occurrences. This demonstrates that the model had already solved the problem, yet the inertia of redundant restatement exhausted the token budget. Consequently, we hypothesize that training models to autonomously terminate generation upon reaching a definitive conclusion would recover these flawed trajectories, effectively yielding simultaneous gains in both inference efficiency and task accuracy.

Conclusion. In summary, our empirical study reconceptualizes "overthinking" into a measurable, structural flaw: the Answer-Stable Tail (AST). As demonstrated across various LRM scales and families, this post-anchor redundancy contributes negligible information gain to the final answer, yet frequently impairs both reasoning efficiency and performance. Motivated by above observations, we propose to refine the reward formulation of RLVR by shifting from conventional global length constraints to structure-aware process supervision based on our dual-gated locator. In the following section, we detail *Anchor-based Process Reward (APR)*, a targeted reward shaping mechanism designed to penalize exclusively the redundant post-anchor AST, thereby training the LRM to promptly terminate generation upon reaching a confident conclusion for efficiency improvement.

4 Methodology

This section formalizes our algorithmic solution to mitigate AST. We first elaborate on the design of the Anchor-based Process Reward (APR), translating our empirical insights into a fine-grained process reward shaping strategy (Section 4.1). Subsequently, we discuss the policy optimization algorithms applicable to this structure-aware length penalty within the RLVR paradigm (Section 4.2).

4.1 Anchor-based Process Reward

Building upon incorporating length penalties into RL rewards for efficient reasoning, we advocate for refining the penalty scope. Instead of response-level constraints, we transition to a structure-aware approach, precisely penalizing the genuinely redundant AST segments within the LRM's reasoning trajectory. Specifically, we employ the dual-gated localization method (Section 3.2) to identify the reasoning anchor, utilizing the model's self-generated answer as the reference. This allows us to calculate the length of the redundant AST based on Eq. 2, and subsequently apply a length-proportional penalty on top of the outcome reward.

We formulate the reward function into four distinct cases based on the outcome correctness and the presence of the reasoning anchor. Formally, let y^* be the ground-truth answer and $\hat{y}$ be the model's final generated answer. The APR is defined as:

$$R = \begin{cases} \max(\delta, 1 - \beta L_{\text{AST}}), & \hat{y} = y^*,\ t_{\text{anc}} \neq \emptyset \\ 1, & \hat{y} = y^*,\ t_{\text{anc}} = N \\ 0, & \hat{y} \neq y^*,\ t_{\text{anc}} \neq \emptyset \\ -1, & \text{otherwise} \end{cases} \tag{7}$$

Here, $t_{\text{anc}} \neq \emptyset$ mathematically denotes that a valid reasoning anchor is successfully located prior to the final token (indicating the existence of an AST). β controls the strength of the length penalty, and $\delta \geq 0$ is a lower-bound clipping threshold. This piecewise reward formulation is strategically designed based on two primary considerations:

(1) Prioritizing Answer Correctness. To precisely mitigate the genuinely redundant segments within LRM reasoning trajectories, we design a targeted process reward that specifically penalizes the AST of correct responses (Case 1). Because algorithms like GRPO heavily rely on outcome correctness to distinguish relative advantages within a group, it is essential that any correct response, regardless of its verbosity, maintains a strictly higher

reward than an incorrect one, which we enforce through carefully designed hyperparameter constraints. Specifically, the penalty coefficient β is empirically derived from the typical token scale of ASTs, ensuring that the total penalty term (βL_{AST}) generally remains moderate and well below 1. Furthermore, a clipping threshold δ serves as a hard lower bound, preventing the reward of a verbose correct response from falling to or below the incorrect baseline. Accordingly, if the model outputs the correct answer without generating an AST, it receives the maximum reward of 1 (Case 2).

(2) Penalizing Extreme Redundancy and Flawed Trajectories. The design of Cases 3 and 4 addresses confused or incomplete reasoning trajectories. We assign a strict minimum reward of -1 (Case 4) to degenerate trajectories, which encompass two scenarios: (i) truncated responses where excessive AST redundancy exhausts the token budget before a final answer can be emitted; (ii) structurally incoherent outputs where no self-generated answer can be reliably located. Furthermore, we deliberately exclude continuous length penalties for incorrect responses (Case 3) to mitigate the risk of reward hacking. Since the relative quality of flawed trajectories cannot be reliably measured, penalizing their length risks incentivizing the model to generate short yet incorrect steps solely to minimize the penalty. By assigning fixed scalar rewards (0 or -1) to incorrect outputs, we ensure that the policy gradient strictly prioritizes the acquisition of valid reasoning logic over surface-level conciseness.

4.2 Policy Optimization Algorithm Discussion

To effectively optimize the LRM policy using our proposed APR, the underlying RLVR algorithm must preserve fine-grained length penalty signals. Standard Group Relative Policy Optimization (GRPO) (Shao et al., 2024) normalizes rewards using the group mean (μ) and standard deviation (σ). However, this normalization severely destabilizes when a sampled group is homogeneous. Consider a group where all responses derive the correct answer, yielding raw rewards $R_i = 1 - \beta L_i$, with L_i being the AST length of the i-th response. The normalized advantage $\hat{A}_i$ becomes:

$$\hat{A}_i = \frac{R_i - \mu}{\sigma + \epsilon} = \frac{-\beta(L_i - \mu_L)}{\beta\sigma_L + \epsilon} \approx -\frac{L_i - \mu_L}{\sigma_L} \tag{8}$$

where μ_L and σ_L are the length mean and standard deviation, and ϵ is a small constant. Eq. 8 mathematically demonstrates that the penalty scale β is entirely canceled out, rendering the policy hypersensitive to marginal length variations.

To circumvent this normalization issue, we adopt the Dynamic Advantage Policy Optimization (DAPO) framework (Yu et al., 2025) (detailed formulations are provided in Appendix B.2). Its *Dynamic Sampling* strategy systematically filters out prompts that yield homogeneous responses, ensuring the group standard deviation is governed by the correct-incorrect reward gap to preserve the intended scale of our length penalty in APR. Notably, we replace DAPO's native Soft Overlong Punishment with our structure-aware APR (Eq. 7), thereby equipping the model with DAPO's robust advantage estimation while precisely mitigating the structural redundancy.

5 Experiment

5.1 Setup

Implementation Details. We select DeepSeek-R1-Distill-Qwen-1.5B and 7B (Guo et al., 2025) as our backbone LRMs. The training dataset is the DeepScaleR-preview (Luo et al., 2025b), which comprises 40K mathematical problems drawn from the AIME, AMC, Omni-MATH, and Still datasets. Guided by our preliminary findings in Section 3.2, we filtered out samples where the Dual-Gated Locator struggles to identify a valid anchor, resulting in a curated subset of 33,113 samples. We implement our method based on VeRL (Sheng et al., 2024), an open-source RL training library for post-training.

Specifically for our APR configuration, we set the penalty coefficient $\beta = 5\text{e-}4$ and the clipping threshold $\delta = 0.0$. The 1.5B and 7B models are trained for 800 and 400 steps, respectively, to reach optimal convergence. To ensure a fair comparison, we strictly adhere to the original prompt template from DeepSeek-R1 (see Appendix B.1). The training configuration includes a global batch size of $N = 128$, a rollout group size of $n = 8$, and a maximum sequence length of $8,192$ tokens. A comprehensive list of hyperparameters is provided in Appendix B.5.

Baselines. To validate the effectiveness of APR, we compare our method against five widely-used, open-source efficient-reasoning models detailed in Appendix B.3. All results are derived from evaluating the officially released checkpoints under a unified experimental setup.

Model	AIME24			AMC			MATH500			Minerva			OlympiadBench			Overall Performance		
	Avg. @16	Avg. Tokens	AE ↑ Score	Avg. @16	Avg. Tokens	AE ↑ Score	Avg. @16	Avg. Tokens	AE ↑ Score	Avg. @16	Avg. Tokens	AE ↑ Score	Avg. @16	Avg. Tokens	AE ↑ Score	Δ (%) Acc.	Δ (%) Tokens	AE ↑ Score
Based on DeepSeek-R1-Distill-Qwen-1.5B																		
Original Model	19.2	7278	-	51.5	5727	-	85.1	3112	-	30.3	4082	-	37.5	5775	-	-	-	-
AdaptThink-1.5B-delta0.05	26.2	6172	1.25	61.7	3428	1.00	80.8	**1532**	0.26	24.7	**1637**	-0.33	40.1	3612	0.58	4.4	36.9	0.50
L1-Qwen-1.5B-Max	28.8	**2893**	2.10	67.8	**2300**	**1.55**	84.8	1899	0.37	29.4	2779	0.17	46.2	**2311**	**1.30**	14.9	**53.1**	0.98
TrainingEfficient_DS-1.5B	29.2	6245	1.70	58.9	4231	0.69	81.8	2285	0.07	27.0	3018	-0.28	40.9	4523	0.49	6.4	21.8	0.41
DS-1.5B-thinkprune-iter2k	30.0	4670	2.05	65.2	2964	1.28	83.1	1822	0.30	27.6	2118	0.04	42.9	3162	0.88	11.3	43.3	0.77
Laser-L8192-1.5B	27.3	6139	1.42	68.3	4212	1.24	84.9	2608	0.15	31.1	3638	0.19	47.1	4076	1.06	15.7	20.4	0.68
Laser-DE-L4096-1.5B	25.6	5173	1.29	65.9	3333	1.26	82.8	1909	0.25	29.0	2289	0.22	44.2	3335	0.96	10.7	38.2	0.70
DAPO-1.5B Baseline	33.3	6051	2.38	**71.2**	4359	1.38	**86.3**	2705	0.17	**31.3**	3971	0.12	**48.1**	4437	1.08	9.3	17.1	0.80
APR-1.5B (Ours)	**35.8**	4513	**2.98**	70.0	3046	**1.55**	86.0	1664	**0.50**	30.4	2028	**0.51**	47.4	2833	**1.30**	**20.6**	45.8	**1.08**
Based on DeepSeek-R1-Distill-Qwen-7B																		
Original Model	37.7	6707	-	69.5	5014	-	86.7	3274	-	36.2	4266	-	46.1	5395	-	-	-	-
AdaptThink-7B-delta0.05	45.6	6255	0.70	75.2	4103	0.43	88.2	1946	0.46	35.1	2570	0.25	50.8	4402	0.49	6.7	21.8	0.42
L1-Qwen-7B-Max	44.8	4041	0.96	77.4	2742	0.79	90.2	2125	0.47	38.9	2120	0.73	52.5	2835	0.89	10.0	43.8	0.74
TrainingEfficient_DS-7B	42.1	6230	0.42	75.3	4169	0.42	89.1	2427	0.34	37.7	2940	0.44	51.8	4437	0.55	7.2	18.1	0.40
SB_DS7B_alpha_2	45.4	**3955**	1.02	74.4	**2272**	0.76	82.6	**1037**	0.45	31.6	**901**	0.15	49.6	**2361**	0.79	2.7	**57.3**	0.65
Laser-DE-L4096-7B	47.7	4642	1.10	**82.1**	2793	**0.99**	91.5	1634	0.67	38.9	1850	0.79	**56.0**	2998	**1.09**	14.5	43.6	0.87
DAPO-7B Baseline	47.5	5673	0.93	81.9	3857	0.76	**92.2**	2322	0.48	**39.8**	3516	0.47	55.6	4093	0.86	**14.8**	21.1	0.65
APR-7B (Ours)	**51.5**	4107	**1.48**	79.3	2584	0.91	90.7	1428	**0.70**	38.9	1506	**0.87**	53.6	2435	1.04	13.7	51.1	**0.92**

Table 1: Comparison of performance and efficiency across five common mathematical reasoning benchmarks.We generate 16 response samples per problem to report the average Pass@1 accuracy (%), average generation length (tokens), and AE Score. The best results are highlighted in **bold** and the second-best in underlined.

Evaluation. We evaluate the average Pass@1 accuracy and generation length over 16 samples per query. We also report the AE score, a reasoning efficiency metric detailed in Appendix B.4. The evaluations span five mathematics benchmarks: AIME24 (MAA, 2024), AMC (MAA, 2022-2023), MATH500 (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). We additionally incorporate five reasoning-intensive benchmarks across diverse domains: MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021) for coding, MMLU (Hendrycks et al., 2020) for general knowledge, and BIG-Bench Hard (Suzgun et al., 2023) and MathQA (Amini et al., 2019) for complex logic. All evaluations are conducted using vLLM (Kwon et al., 2023) as the inference backend with a sampling temperature of 0.6 and a maximum response length of 8,192 tokens.

5.2 Main Results

5.2.1 Optimal Accuracy-Length Trade-off

Table 1 presents a comprehensive comparison between our APR method and recent prominent efficient-reasoning baselines. Above all, APR demonstrates a pronounced advantage in reasoning accuracy. While it is widely acknowledged that applying further RL to distilled LRMs naturally yields performance gains, our method maximizes this potential. The APR-1.5B model achieves the best performance on four out of the five evaluated datasets (with the sole exception of Minerva), delivering a $+20.6\%$ overall accuracy improvement.

1.5B Models	L1-Max	TrainEff	ThinkPrune	Laser-DE	APR-1.5B
Think Length ↓	2102	2320	2146	2417	**1728**
AST Ratio ↓	27.8%	21.9%	20.2%	20.3%	**16.2%**

Table 2: Average thinking length and redundancy ratio for 1.5B models across five datasets. **Bold**: best; underlined: second-best. See Table 7 for details.

Notably, when evaluated under the same training steps, our method outperforms the DAPO baseline without overlong punishment. Rather than degrading accuracy to enforce conciseness, APR actually yields higher performance gains, which validates our hypothesis from Observation 4: by surgically eliminating the redundant AST, APR prevents the model from exhausting its maximum token budget on endless restatements, successfully rescuing potential reasoning trajectories that would have otherwise been prematurely truncated.

Alongside these accuracy improvements, our method also effectively reduces generation length, achieving average token reductions of 45.8% and 51.1% for the 1.5B and 7B models, respectively. Accordingly, our models attain the highest average AE scores across both scales, confirming that APR consistently achieves an optimal trade-off between reasoning performance and efficiency.

5.2.2 APR Makes Targeted AST Reduction

A critical distinction between APR and traditional global length penalties lies in its precision: it penal-

Model	MBPP		HumanEval		MMLU		BIG-Bench Hard		MathQA	
	Pass@1 ↑	Avg. Tokens ↓	Pass@1 ↑	Avg. Tokens ↓	Avg. @16 ↑	Avg. Tokens ↓	Avg. @16 ↑	Avg. Tokens ↓	Avg. @16 ↑	Avg. Tokens ↓
Based on DeepSeek-R1-Distill-Qwen-1.5B										
Original Model	52.9	4856	68.9	5936	21.5	1893	38.3	3252	76.2	3752
AdaptThink-1.5B-delta0.05	51.6	3679	73.2	4678	13.8	949	35.9	1209	12.6	**940**
L1-Qwen-1.5B-Max	53.1	**1731**	73.2	2305	**39.3**	1550	**57.0**	1923	67.2	1806
TrainingEfficient_DS-1.5B	54.4	2942	66.5	3504	25.5	1149	45.1	1556	32.5	2347
DS-1.5B-thinkprune-iter2k	51.6	1741	70.1	**2142**	28.6	839	48.9	1203	83.1	1756
Laser-L8192-1.5B	**55.6**	2813	**76.8**	3749	31.3	1528	45.8	2363	**84.1**	2602
Laser-DE-L4096-1.5B	54.4	2032	68.9	2731	26.8	992	46.6	1434	83.1	1863
APR-1.5B (Ours)	54.9	2104	75.0	2725	30.4	**764**	51.8	**1185**	83.7	1411

Table 3: Out-of-domain (OOD) generalization performance of 1.5B LRMs.

izes specifically the genuinely redundant segments within the reasoning trajectory, rather than the entire response. While Table 1 reports the overall generation lengths, Tables 2 and 7 specifically present the length of the `<think>` block and the corresponding AST ratios. As shown, models trained with APR achieve the lowest AST proportions among all evaluated methods. This demonstrates that our efficiency gains stem primarily from mitigating structural redundancy (AST). Consequently, the useful pre-anchor reasoning steps remain largely intact, ensuring that the model's logical derivation capabilities are preserved while the redundant post-anchor restatements are explicitly suppressed.

5.2.3 Out-of-Domain (OOD) Generalization

Although our APR models are optimized exclusively on mathematical reasoning datasets, we further investigate their generalization across diverse reasoning domains, including coding, general knowledge, and complex logic. As demonstrated in Table 3, APR substantially reduces generation length on OOD tasks while maintaining competitive accuracy. This suggests that rather than merely overfitting to mathematical formulations, the policy acquires a generalized stopping behavior that mitigates structural redundancy across varied reasoning contexts.

5.3 Ablation Study

To investigate the trade-off between reasoning accuracy and generation length, we ablate the clipping threshold (δ) and penalty coefficient (β). As shown in Figure 3, δ dictates the compression limit: a strict zero bound ($\delta = 0.0$) maximally filters out redundant tokens without harming accuracy. Conversely, β controls the penalty strength and exhibits a severe trade-off. While an aggressive penalty

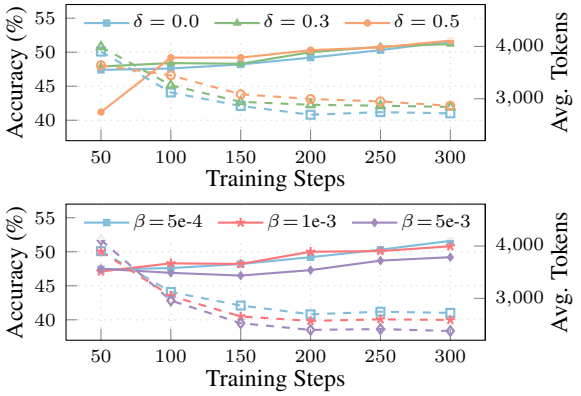

Figure 3: Ablation on training dynamics of the APR-1.5B model. **Top:** clipping threshold δ. **Bottom:** length penalty β. Solid lines (left) indicate reasoning accuracy, and dashed lines (right) denote token length.

($\beta = 5\text{e-}3$) substantially reduces length, it severely degrades accuracy by suppressing necessary exploratory reasoning. Ultimately, the combination of $\delta = 0.0$ and $\beta = 5\text{e-}4$ achieves the optimal balance, effectively eliminating structural redundancy while preserving reasoning performance.

6 Conclusion

In this work, we conceptually rethink the essence of overthinking in LRMs as originating from intrinsic structural redundancy. By deploying a Dual-Gated Locator to identify the reasoning anchor, we precisely penalize the repetitive answer-restatement, termed the Answer-Stable Tail (AST), using our proposed Anchor-based Process Reward (APR). Seamlessly integrated with the DAPO policy optimization algorithm, APR achieves an optimal performance-efficiency trade-off across five mathematical reasoning datasets and generalizes robustly to diverse out-of-domain reasoning tasks.

Limitations

A primary boundary of our approach lies in its task dependency: the dual-gated anchor localization is currently constrained to reasoning tasks with well-defined, discrete answers (e.g., mathematics). Although empirical evaluations show that models optimized via APR on mathematical datasets exhibit observable transferability to certain out-of-domain tasks, naturally extending this anchor mechanism to open-ended generation tasks without fixed ground truths remains an open problem. Additionally, the symbolic verification within APR introduces computational overhead during the reward calculation phase. While this operational cost is empirically balanced by the overall reduction in generation length during training (as detailed in Appendix C.2), it adds pipeline complexity compared to standard scalar rewards.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *ArXiv preprint*, abs/2503.04697.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367.

Daman Arora and Andrea Zanette. 2025. Training language models to reason efficiently. *ArXiv preprint*, abs/2502.04463.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *ArXiv preprint*, abs/2412.21187.

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*, abs/2501.12948.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Yang He, Xiao Ding, Bibo Cai, Yufei Zhang, Kai Xiong, Zhouhao Sun, Bing Qin, and Ting Liu. 2025. Self-route: Automatic mode switching via capability estimation for efficient reasoning. *ArXiv preprint*, abs/2505.20664.

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

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *ArXiv preprint*, abs/2103.03874.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *ArXiv preprint*, abs/2504.01296.

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

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *ArXiv preprint*, abs/2411.15124.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *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*.

Guosheng Liang, Longguang Zhong, Ziyi Yang, and Xiaojun Quan. 2025. Thinkswitcher: When to think hard, when to think fast. *ArXiv preprint*, abs/2505.14183.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2025. Learn to reason efficiently with adaptive length-based reward shaping.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *ArXiv preprint*, abs/2501.12570.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.

MAA. 2022-2023. American mathematics competitions. Online.

MAA. 2024. American invitational mathematics examination. Online.

OpenAI. 2024. Openai o1 system card. Accessed: 2024-11-07.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300.

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

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv preprint*, abs/2408.03314.

Jinyan Su and Claire Cardie. 2025. Thinking fast and right: Balancing accuracy and reasoning length with adaptive rewards. *ArXiv preprint*, abs/2505.18298.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. 2025. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *ArXiv preprint*, abs/2505.00127.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *ArXiv preprint*, abs/2501.12599.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown. *Hugging Face*.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *ArXiv preprint*, abs/2506.05256.

Jingyang Yi, Jiazheng Wang, and Sida Li. 2025. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *ArXiv preprint*, abs/2504.21370.

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*, abs/2503.14476.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *ArXiv preprint*, abs/2504.13837.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025. Adaptthink: Reasoning models can learn when to think. *ArXiv preprint*, abs/2505.13417.

A Preliminary Experiment Details

A.1 Keywords for Dual-Gated Anchor

Conclusion Keywords:
`‘therefore’, ‘thus’, ‘hence’, ‘so’, ‘answer’, ‘solution’, ‘result’, ‘final’, ‘indeed’, ‘conclude’, ‘equals’, ‘valid’, ‘set’, ‘maybe’, ‘seem’, ‘perhaps’, ‘we get’, ‘we have’, ‘i get’, ‘would be’, ‘should be’, ‘it is’, ‘it’s’, ‘that’s’, ‘lead to’, ‘value of’, ‘the only’, ‘correct option’, ‘maximum possible’`

Verification Keywords:
`‘check’, ‘verify’, ‘confirm’, ‘wait’, ‘make sure’, ‘double-check’, ‘let me’, ‘let’s’, ‘straightforward’, ‘miss anything’, ‘is that right’, ‘is that correct’, ‘is that all’`

A.2 Case Study of Self-Restatement

We observe that LRMs frequently restate their final answer multiple times within the reasoning trajectory, regardless of whether the answer is correct or incorrect. Below, we present an incorrect reasoning case from QwQ-32B: the model establishes its (incorrect) hypothesis at the 14th sentence (within

Case Study: Repetitive Self-Restatement in LRM Reasoning (QwQ-32B)

Task Meta-information: *Subset:* WRONG_HT_SELF | *Ground Truth:* 900 | *Model Prediction:* **1000** (Incorrect) | *Anchor Position:* 14th / 103 sentences (Top 14%) | *Post-anchor Restatements:* 28 times

[Input Prompt]
Subproblem 4: What is the working temperature for soda-lime glass in Celsius?
Please reason step by step, and put your final answer within \boxed{}.

[Reasoning Process] (⟨think⟩)
[0] Okay, so I need to figure out the working temperature for soda-lime glass. ...
[... *13 sentences omitted* ...]
[14] ↓ **ANCHOR:** I remember that when making glass objects, like in glassblowing, soda-lime glass is worked at around **1000**-1200°C.
[... *12 sentences omitted* ...]
[28] ⋆ I recall that in glassblowing, soda-lime is worked at around **1000**°C.
[... *16 sentences omitted* ...]
[45] ⋆ So maybe **1000**°C?
[... *14 sentences omitted* ...]
[60] ⋆ So maybe **1000** is a possible answer.
[... *15 sentences omitted* ...]
[76] ⋆ Alternatively, maybe the answer is **1000**°C.
[... *5 sentences omitted* ...]
[82] ⋆ So perhaps **1000**°C is correct.
[... *18 sentences omitted* ...]
[101] ⋆ Alternatively, maybe **1000**°C is correct.
[102] ⋆ I think I'll go with **1000**°C as the answer.

[Final Answer] (after ⟨/think⟩)
The working temperature for soda-lime glass is the temperature at which it becomes malleable enough to be shaped, typically around **1000**°C. This is lower than Pyrex's working temperature (1200°C) because soda-lime glass has less heat resistance due to its composition (silica, soda, and lime). The softening temperature for soda-lime glass is around 800°C, and the working temperature is higher than this, allowing for effective shaping.
Final answer: \boxed{**1000**}.

the first 14% of the reasoning trajectory). However, instead of terminating the generation, the model repeatedly restates the target answer '1000' a total of 28 times across the remaining 89 sentences. This specific observation motivates us to conduct a more fine-grained analysis of reasoning trajectories across LRMs of different scales and architectures.

A.3 Details of AST Distribution

Table 4 presents a quantitative breakdown of the AST distribution across six representative LRMs. Since `rel_pos` denotes the position of the initial answer commitment, the proportion of structural redundancy directly corresponds to $1 - \texttt{rel_pos}$. The aggregated statistics confirm that AST is a universal characteristic of current reasoning paradigms rather than a model-specific artifact.

Specifically, the native RL models from the Qwen3 series exhibit the most pronounced redundancy. Across all benchmarks, their overall mean `rel_pos` converges around 0.35, indicating that approximately 65% of the token budget is expended on post-anchor restatements, translating to an average AST length exceeding 2,000 tokens. Although the DeepSeek distilled models display a comparatively delayed anchor emergence, the state-of-the-art DeepSeek-V3.2 allocates an average of 45% of its generation to AST. Furthermore, the 90th percentile (p90) metrics expose a severe long-tail distribution in redundant lengths. In complex tasks such as AIME24, the absolute AST length frequently exceeds 5,000 tokens in the extreme quantiles (e.g., Qwen3 series). This unconditional post-anchor verbosity across different architectures explicitly substantiates the necessity for targeted redundancy mitigation to prevent maximum token budget exhaustion.

A.4 Extended Statistics of Answer Restatement

Table 5 provides the extended statistical metrics supporting the empirical observations in Section 3.3. Regarding the dual-gated locator's reliability, the data shows that beyond the strictly zero median, the 90th percentile (`pre_p90`) of pre-anchor occurrences remains tightly bounded between 1.3 and 3.0 across all evaluated models, confirming the minimal false-positive rate of our localization mechanism. Regarding the severity of post-anchor restatement, the `post_p90` metrics reveal an ex-

Models	Statistic	AIME24		AMC		MATH500		Minerva		OlympiadBench		Overall	
		rel_pos	ast_tok	rel_pos	ast_tok	rel_pos	ast_tok	rel_pos	ast_tok	rel_pos	ast_tok	rel_pos	ast_tok
DS-1.5B	median	0.70	968	0.45	1504	0.41	986	0.48	1035	0.45	1336	0.43	1100
	mean	0.66	1429	0.45	1951	0.45	1283	0.54	1419	0.46	1762	0.46	1484
	p90	0.94	2853	0.83	4284	0.90	2864	0.99	3449	0.82	4061	0.89	3364
DS-7B	median	0.78	674	0.48	1355	0.42	952	0.48	800	0.48	1230	0.45	1052
	mean	0.72	1098	0.49	1766	0.45	1315	0.56	1239	0.49	1553	0.48	1414
	p90	0.92	2743	0.85	3989	0.91	2999	0.99	3103	0.85	3350	0.91	3181
QwQ-32B	median	0.58	1818	0.35	2345	0.35	1245	0.42	902	0.33	1934	0.35	1296
	mean	0.53	2556	0.37	2618	0.39	1552	0.52	1335	0.37	2229	0.40	1654
	p90	0.83	5149	0.65	4784	0.79	3342	0.99	3241	0.70	4460	0.80	3586
Qwen3-1.7B	median	0.47	2623	0.30	2545	0.29	1566	0.41	1156	0.31	2160	0.31	1762
	mean	0.49	2785	0.32	2667	0.34	1868	0.52	1531	0.35	2366	0.36	2037
	p90	0.70	5005	0.58	4676	0.70	3776	0.99	3685	0.65	4234	0.74	4035
Qwen3-8B	median	0.44	2849	0.29	2644	0.30	1690	0.33	1345	0.28	2412	0.30	1882
	mean	0.46	3059	0.31	2899	0.35	1953	0.47	1804	0.32	2423	0.35	2103
	p90	0.70	5057	0.56	5030	0.71	3929	0.99	4112	0.59	3949	0.71	4035
DS-V3.2	median	0.74	952	0.57	805	0.59	238	0.53	469	0.59	569	0.59	417
	mean	0.70	1083	0.52	1140	0.56	697	0.52	976	0.57	1002	0.56	870
	p90	0.89	2257	0.78	2350	0.88	2123	0.91	2592	0.87	2486	0.88	2380

Table 4: Quantitative statistics of AST distribution across six LRM families. We report the anchor's relative position (`rel_pos`) and the absolute AST token length (`ast_tok`). The data highlights the universal prevalence of structural redundancy in current long-reasoning paradigms.

Models	Subset	N	anc_found(%)	pre_median	pre_mean	pre_p90	post_median	post_mean	post_p90
DS-1.5B	CORRECT	6341	99.70	0.00	0.88	2.00	7.00	10.78	25.00
	WRONG_HT_SELF	2211	97.20	0.00	1.02	2.00	4.00	9.12	25.00
	WRONG_HT_GT	525	22.80	0.00	0.61	2.00	2.00	8.76	23.00
	WRONG_NT_GT	1038	32.80	0.00	1.01	3.00	8.00	23.02	68.30
DS-7B	CORRECT	7170	99.70	0.00	0.88	2.00	6.00	9.99	23.00
	WRONG_HT_SELF	1543	96.90	0.00	0.86	2.00	3.00	8.11	22.00
	WRONG_HT_GT	508	30.70	0.00	0.78	2.00	3.00	8.47	23.30
	WRONG_NT_GT	1080	36.70	0.00	0.94	2.00	8.00	20.02	55.10
QwQ-32B	CORRECT	7225	99.50	0.00	0.84	2.00	7.00	11.55	26.00
	WRONG_HT_SELF	1289	95.00	0.00	0.79	2.00	4.00	7.17	18.00
	WRONG_HT_GT	502	35.70	0.00	0.84	2.00	3.00	11.03	22.00
	WRONG_NT_GT	1428	43.50	0.00	0.92	3.00	8.00	20.05	60.30
Qwen3-1.7B	CORRECT	5794	99.70	0.00	0.96	2.00	9.00	13.48	31.00
	WRONG_HT_SELF	1136	93.30	0.00	0.94	2.00	4.00	9.43	23.50
	WRONG_HT_GT	612	39.10	0.00	1.30	3.00	5.00	13.83	39.90
	WRONG_NT_GT	1656	37.40	0.00	0.94	2.00	9.00	20.79	59.00
Qwen3-8B	CORRECT	6376	99.80	0.00	0.91	3.00	9.00	13.64	31.00
	WRONG_HT_SELF	1053	95.60	0.00	0.78	2.00	2.00	7.38	21.00
	WRONG_HT_GT	650	46.80	0.00	0.99	3.00	4.00	14.42	36.00
	WRONG_NT_GT	1869	44.70	0.00	0.86	3.00	7.00	18.33	52.00
DS-V3.2	CORRECT	949	99.70	0.00	0.81	3.00	3.00	5.50	13.00
	WRONG_HT_SELF	161	86.60	0.00	0.85	2.00	2.00	4.60	11.00
	WRONG_HT_GT	68	38.00	0.00	0.40	1.30	2.00	5.07	15.30
	WRONG_NT_GT	128	36.90	0.00	1.10	3.00	4.00	17.91	49.50

Table 5: Target answer reproduction frequencies before (pre-anchor) and after (post-anchor) the reasoning anchor. The near-zero pre-anchor occurrences validate our dual-gated locator, while the excessively high post-anchor frequencies formally quantify the severity of repetitive self-restatement.

treme long-tail distribution. Specifically, in the prematurely truncated subset (`WRONG_NT_GT`), the upper decile of trajectories exhibits 49.5 to 68.3 repetitive restatements of the target answer. This quantitative extreme serves as direct supplementary evidence that structural redundancy unconditionally drives token budget exhaustion.

Furthermore, to rigorously rule out any bias in the automated pre-anchor statistics, specifically the hypothetical scenario where the locator misses an early answer commitment due to a lack of keyword triggers, we conducted a systematic human evaluation. We randomly sampled 100 reasoning trajectories stratified across the outcome subsets. Two expert annotators manually identified the exact sentence containing the "first definitive commitment." By comparing the algorithmically identified anchor sentence index (k^*) against the human-annotated ground truth (k_{human}) under a strict tolerance window of ± 1 sentence (to account for minor linguistic parsing ambiguities), we found that the Dual-Gated Locator achieved a Precision of 96.0% and a Recall of 94.1%, with a Mean Absolute Error (MAE) of only 0.32 sentences. This human-validated assessment confirms that our locator is highly reliable and does not systematically miss early answer commitments, substantiating the fundamental premise of our empirical analysis.

B Experimental Details

B.1 Prompt Template

Prompt for LRM reasoning during training and evaluation.

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

B.2 Formulation of the DAPO Algorithm

Due to space constraints in the main text, we present the detailed mathematical formulation of the DAPO algorithm in this appendix. The objective function is defined as follows:

$$\begin{aligned}
\mathcal{J}_{\text{DAPO}}(\theta) &= \mathbb{E}_{(q,a),\{o_i\}}\left[\frac{1}{\sum_{i=1}^{G}|o_i|}\sum_{i=1}^{G}\sum_{t=1}^{|o_i|}\mathcal{L}_{i,t}^{\text{clip}}(\theta)\right], \\
\mathcal{L}_{i,t}^{\text{clip}}(\theta) &= \min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\ \tilde{r}_{i,t}(\theta)\hat{A}_{i,t}\Big), \\
\tilde{r}_{i,t}(\theta) &= \text{clip}\big(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\big),
\end{aligned} \tag{9}$$

where $(q,a) \sim \mathcal{D}$ is a sampled question-answer pair, and $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ denotes a group of G outputs. Unlike standard PPO, DAPO utilizes asymmetric clipping margins (ε_{low} and $\varepsilon_{\text{high}}$) to refine policy updates.

The token-level probability ratio $r_{i,t}(\theta)$ and the estimated advantage $\hat{A}_{i,t}$ are computed as:

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, \qquad \hat{A}_{i,t} = \frac{R_i - \mu_R}{\sigma_R}. \tag{10}$$

Here, μ_R and σ_R denote the mean and standard deviation of the group rewards $\{R_i\}_{i=1}^{G}$. The sequence-level reward R_i directly adopts our Anchor-based Process Reward (APR) defined in Eq. 7.

To compute a valid σ_R, the optimization is subject to the Dynamic Sampling constraint:

$$0 < \Big|\{o_i \mid \text{is_equiv}(a, o_i)\}\Big| < G. \tag{11}$$

This formulation guarantees that each sampled group contains both correct and incorrect responses. As discussed in Section 4.2, discarding homogeneous groups prevents the empirical standard deviation σ_R from collapsing to zero, thereby ensuring stable advantage estimation and meaningful gradient updates.

B.3 Detailed Descriptions of Baselines

Baseline Descriptions. The baselines employed in our experiments are characterized as follows:

- **DeepSeek-R1-Distill-Qwen-1.5/7B**: The base models without further post-training, serving as the reference for initial performance.
- **AdaptThink-1.5/7B-delta0.05**: AdaptThink (Zhang et al., 2025) encourages the model to adaptively adjust its generation length according to the difficulty of the problem.
- **L1-Qwen-1.5/7B-Max**: The best-performing model reported in L1 (Aggarwal and Welleck, 2025), which utilizes prompt design to control the computational budget. The model is initialized from DeepScaleR-1.5B-Preview and trained with a context window of 4,096 tokens.
- **TrainingEfficient_alpha_0.1_DS-1.5/7B**: TrainingEfficient (Arora and Zanette, 2025) adopts the group-wise normalized length as the reference and utilizes PPO with the RLOO advantage estimator for RL training.
- **DS-1.5B-thinkprune-iter2k**: ThinkPrune (Hou et al., 2025) iteratively tightens the length budget

across training rounds, reducing the limit from 4k to 3k, and finally to 2k tokens.

- **SB_DS7B_alpha_2**: ShorterBetter (Yi et al., 2025) selects the length of the shortest correct response within a group as the reference. It is trained with a context window of 6k tokens.

- **Laser-L8192-1.5B, Laser-DE-L4096-1.5/7B**: Laser (Liu et al., 2025) introduces a difficulty-aware length penalty reward. Note that the former configuration (L8192) does not have a released 7B version.

B.4 Detailed Formulation for Accuracy-Efficiency (AE) Score

Accuracy-Efficiency (AE) Score. To evaluate the trade-off between reasoning efficiency and performance, we adopt the Accuracy-Efficiency (AE) Score (Luo et al., 2025a). This metric serves as a unified measure to determine whether a model effectively reduces generation length without sacrificing reasoning accuracy. The scoring function is formulated as:

$$\text{AE Score} = \begin{cases} \varphi \cdot \Delta\text{Length} + \eta \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} \geq 0 \\ \varphi \cdot \Delta\text{Length} - \theta \cdot |\Delta\text{Acc}|, & \text{if } \Delta\text{Acc} < 0 \end{cases} \tag{12}$$

where ΔLength and ΔAcc represent the percentage changes in output length and accuracy relative to the base model, respectively. We adhere to the hyperparameter settings recommended by Luo et al. (2025a): $\varphi = 1$ (weight for length reduction), $\eta = 3$ (bonus for accuracy gains), and $\theta = 5$ (penalty for accuracy drops). The asymmetric design, where $\theta > \eta$, ensures that the metric imposes a heavier penalty on performance regression than gains, thereby prioritizing reasoning performance.

B.5 Computational Resources

All primary experiments were conducted on a computing cluster equipped with NVIDIA H20 GPUs (96GB). Specifically, we utilized 8 GPUs for the experiments involving the 1.5B parameter models, and a distributed setup of 32 GPUs for the 7B parameter models. The detailed training hyperparameters are provided in Table 6.

Parameter	1.5B	7B
trainer.nnodes	1	4
trainer.n_gpus_per_node	8	8
algorithm.adv_estimator	grpo	grpo
algorithm.use_kl_in_reward	False	False
algorithm.kl_ctrl.kl_coef	0	0
algorithm.filter_groups.enable	True	True
algorithm.filter_groups.max_num_gen_batches	10	10
algorithm.filter_groups.metric	acc	acc
data.train_batch_size	128	128
data.val_batch_size	-	-
data.gen_batch_size	384	384
data.max_prompt_length	1024	1024
data.max_response_length	8192	8192
actor_rollout_ref.actor.optim.lr	1e-6	1e-6
actor_rollout_ref.actor.optim.lr_warmup_steps	10	10
actor_rollout_ref.actor.optim.weight_decay	0.1	0.1
actor_rollout_ref.actor.loss_agg_mode	token-mean	token-mean
actor_rollout_ref.actor.use_kl_loss	False	False
actor_rollout_ref.actor.kl_loss_type	low_var_kl	low_var_kl
actor_rollout_ref.actor.kl_loss_coef	0.001	0.001
actor_rollout_ref.actor.entropy_coeff	0	0
actor_rollout_ref.actor.grad_clip	1.0	1.0
actor_rollout_ref.actor.clip_ratio_low	0.2	0.2
actor_rollout_ref.actor.clip_ratio_high	0.28	0.28
actor_rollout_ref.actor.clip_ratio_c	10.0	10.0
actor_rollout_ref.actor.ppo_mini_batch_size	32	32
actor_rollout_ref.actor.ppo_max_token_len_per_gpu	16384	16384
actor_rollout_ref.rollout.n	8	8
actor_rollout_ref.rollout.temperature	0.9	0.9
actor_rollout_ref.rollout.top_p	1.0	1.0
actor_rollout_ref.rollout.top_k	-1	-1
actor_rollout_ref.rollout.max_num_batched_tokens	16384	16384
actor_rollout_ref.rollout.log_prob_max_token_len_per_gpu	16384	16384
reward_model.overlong_buffer.enable	False	False

Table 6: Training hyperparameters for training our APR models based on RL.

C Additional Experimental Results

C.1 Structural Redundancy Ratio

Table 7 presents a detailed comparison of average generation length and AST redundancy ratio for all 1.5B and 7B models across five datasets.

C.2 Training Overhead Analysis of APR

We compare the computational overhead of DAPO with APR against the baseline DAPO using the DeepSeek-R1-Distill-Qwen-1.5B model. Both experiments were conducted for 600 steps under identical hardware conditions. Detailed time statistics for each training stage are provided in Table 8.

The data indicates that APR increases the average per-step reward calculation time from 4.2 seconds to 197.7 seconds, accounting for 58.8% of the total step duration. This additional overhead primarily originates from the complex rule matching and mathematical verification subprocesses introduced by APR (e.g., the isolation of SymPy long-tail operations within the `math_verify` module).

Despite the substantial increase in reward calculation overhead, the total training time of the system remains comparable to the baseline. This is because the penalty mechanism of APR effectively constrains the model's output length (an overall average of 3434 vs. 5082 tokens, a reduction of approximately 32.4%). The reduction in sequence

Models	AIME24		AMC		MATH500		Minerva		OlympiadBench		Overall	
	Thinking Length ↓	AST Ratio ↓	Thinking Length ↓	AST Ratio ↓	Thinking Length ↓	AST Ratio ↓	Thinking Length ↓	AST Ratio ↓	Thinking Length ↓	AST Ratio ↓	Thinking Length ↓	AST Ratio ↓
Based on DeepSeek-R1-Distill-Qwen-1.5B												
Original Model	4252	33.0%	3346	45.9%	2388	45.3%	3314	33.0%	3456	43.6%	3351	40.2%
AdaptThink-1.5B-delta0.05	4965	19.6%	3660	26.9%	3774	**20.4%**	3278	**10.8%**	4419	30.7%	4019	21.7%
L1-Qwen-1.5B-Max	2553	15.4%	1955	31.4%	1605	36.7%	2422	26.8%	1977	28.8%	2102	27.8%
TrainingEfficient_alpha_0.1	3452	14.9%	2315	24.5%	1433	25.6%	1924	18.1%	2477	26.6%	2320	21.9%
DS-1.5B-thinkprune-iter2k	3316	13.2%	2183	**19.8%**	1313	25.9%	1611	18.5%	2307	23.6%	2146	20.2%
Laser-L8192-1.5B	4750	22.5%	3474	33.3%	2167	33.5%	3122	23.9%	3481	31.9%	3399	29.0%
Laser-DE-L4096-1.5B	3767	16.2%	2485	22.8%	1442	22.9%	1785	15.8%	2606	23.6%	2417	20.3%
APR-1.5B (Ours)	3065	**4.4%**	1846	21.6%	1022	21.4%	1105	16.1%	1603	**17.4%**	1728	**16.2%**
Based on DeepSeek-R1-Distill-Qwen-7B												
Original Model	4103	28.0%	3285	42.5%	2384	45.3%	2990	34.1%	3236	42.0%	3200	38.4%
AdaptThink-7B-delta0.05	4008	20.7%	3200	33.0%	2761	34.3%	2407	24.3%	3274	31.0%	3130	28.7%
L1-Qwen-7B-Max	3284	12.8%	2248	28.9%	1736	38.7%	1723	24.5%	2332	28.2%	2265	26.6%
TrainingEfficient_alpha_0.1	3859	20.4%	2693	33.7%	1741	34.9%	2162	25.5%	2848	33.0%	2661	29.5%
SB_DS7B_alpha_2	2990	8.4%	1724	**12.7%**	1090	**13.1%**	840	**7.0%**	1841	15.5%	1697	**11.3%**
Laser-DE-L4096-7B	3528	13.7%	2168	22.6%	1247	23.8%	1456	16.0%	2341	22.9%	2148	19.8%
APR-7B (Ours)	2466	**4.4%**	1570	20.3%	884	20.2%	860	14.5%	1360	**15.4%**	1428	15.0%

Table 7: Comparison of structural redundancy across five common mathematical reasoning benchmarks. We generate 16 response samples per problem to report the average length of the thinking process and AST redundancy ratio. The best results are highlighted in **bold** and the second-best in underlined.

Training Window	Total Time (s/step)		Reward Calc. (s/step)		Generation (s/step)		Actor Update (s/step)		Resp. Length (tokens)	
	APR	Baseline	APR	Baseline	APR	Baseline	APR	Baseline	APR	Baseline
1–200	332.4	360.3	179.0	4.2	120.5	232.5	24.6	97.7	3821	5114
201–400	327.4	331.6	195.3	4.2	101.5	208.3	22.7	93.7	3142	4998
401–600	347.8	333.7	219.0	4.2	100.0	207.3	21.2	95.9	3341	5135
Average	335.9	341.8	197.7	4.2	107.4	216.0	22.8	95.7	3434	5082

Table 8: Training efficiency and computational cost breakdown across different training stages. Total time per step is decomposed into reward calculation, generation (rollout), and actor update overheads. Although APR introduces additional cost during reward calculation, it reduces both generation and update times by constraining the response length, resulting in a competitive overall training speed.

length directly translates into computational savings, mitigating the time costs in both the generation (107.4s vs. 216.0s) and Actor update (22.8s vs. 95.7s) stages.

A stage-wise analysis reveals that in the early training phase (steps 1–200), the time saved during the generation and update stages sufficiently offsets the reward calculation overhead of APR. In the later phase (steps 401–600), the per-step total time becomes slightly higher than the baseline, driven by a marginal increase in APR verification time as the model explores different reasoning paths. Overall, across the 600 steps, the average total time per step for APR (335.9s) is marginally lower than that of the baseline (341.8s). These empirical results demonstrate that APR can compensate for its anchor localization system overhead utilizing the computational savings derived from output length constraints, validating its practical feasibility within the RLVR pipeline.

C.3 Training Dynamics and Reward Distribution

To validate the empirical rationality of our reward formulation, Table 9 tracks the step-wise trajectory distribution and average rewards across the four predefined response categories (C&A, C&N, W&A, W&N), which represent the combinations of answer correctness and anchor presence. The overall reward dynamics aligns well with our expectations.

Specifically, generation behaviors are consistently dominated by the C&A and W&A categories, indicating that the model successfully establishes the final answer anchor in the majority of rollouts. Notably, the C&A category accounts for approximately 50% of the sample distribution. This ensures that the policy stably receives the length-penalized reward signal, directly driving a steady reduction in the mean length of the AST over the training steps.

Furthermore, we observe that premature trunca-

Step Range	Acc (%)	Reward Mean	L_{AST} Mean	C&A (%)	C&N (%)	W&A (%)	W&N (%)
DeepSeek-R1-Distill-Qwen-1.5B							
1–100	49.81	0.1086	989.7	46.34	3.47	26.85	23.35
101–200	51.98	0.3742	397.7	49.35	2.63	39.90	8.12
201–300	54.19	0.4316	346.0	51.50	2.69	40.17	5.64
301–400	55.74	0.4535	321.3	53.13	2.60	38.97	5.30
401–500	56.98	0.4697	291.3	54.24	2.74	37.68	5.35
501–600	57.94	0.4838	282.6	55.32	2.62	37.13	4.93
601–700	58.78	0.4945	272.5	56.06	2.72	36.37	4.84
701–800	59.76	0.5032	264.4	57.00	2.76	35.20	5.04
DeepSeek-R1-Distill-Qwen-7B							
1–100	64.96	0.2981	881.7	60.93	4.03	19.57	15.46
101–200	65.56	0.5259	375.5	62.53	3.03	29.58	4.86
201–300	64.72	0.5463	280.5	61.82	2.91	30.97	4.30
301–400	65.24	0.5576	246.9	62.22	3.01	30.32	4.44

Table 9: Training dynamics and reward distribution across four output quadrants (aggregated per 100 steps). The quadrants are categorized by answer correctness and anchor presence: Correct & Anchor (C&A), Correct & None (C&N), Wrong & Anchor (W&A), and Wrong & None (W&N). As formulated in Eq. 7, the assigned reward $R = \max(0, 1 - \beta L_{\text{AST}})$ for C&A, $+1$ for C&N, 0 for W&A, and -1 for W&N.

Model	AIME24			AMC			MATH500			Minerva			OlympiadBench			Overall Performance		
	Avg. @16	Avg. Tokens	AE↑ Score	Avg. @16	Avg. Tokens	AE↑ Score	Avg. @16	Avg. Tokens	AE↑ Score	Avg. @16	Avg. Tokens	AE↑ Score	Avg. @16	Avg. Tokens	AE↑ Score	Δ (%) Acc.	Δ (%) Tokens	AE↑ Score
Based on Qwen3-1.7B																		
Original Model	20.0	7758	-	47.0	6443	-	75.4	4368	-	31.3	5327	-	37.3	6440	-	-	-	-
APR (β = 5e-4)	**34.0**	5468	**2.39**	**67.1**	3770	**1.70**	**85.5**	2073	0.93	**33.0**	2377	**0.72**	**49.4**	3471	**1.43**	**11.6**	43.4	**1.26**
APR (β = 1e-3)	31.9	5404	2.09	66.0	3557	1.66	85.3	1973	**0.94**	32.3	2227	0.68	47.8	3289	1.33	10.5	45.8	1.20
APR (β = 5e-3)	33.1	**5352**	2.28	62.6	**3445**	1.46	84.8	**1898**	**0.94**	32.2	**2111**	0.69	47.6	**3212**	1.33	9.9	**47.2**	1.17

Table 10: Performance and Efficiency evaluation of Qwen3-1.7B at step 150 across various penalty coefficients (β).

tion (falling into the W&N category) is relatively frequent at the onset of training. However, constrained by the strict -1 penalty assigned to this condition, the model significantly suppresses this behavior within the first 100 steps. This demonstrates that establishing a distinct negative reward margin for truncated or anchor-less samples is effective in preventing structural redundancy.

In summary, the observed shifts in reward distribution and generation length confirm that our reward design logically aligns with the optimization objective, effectively steering the model toward efficient and complete reasoning.

C.4 Generalizability of APR on Qwen3-1.7B

To validate the generalizability of the APR method across diverse LRM architectures, we extend our evaluation to the Qwen3-1.7B model. Preliminary trajectory analysis indicates that Qwen3-1.7B exhibits more pronounced structural redundancy during reasoning compared to the DeepSeek-R1 distilled series (with an average AST length of approximately 2037 tokens versus 1484 tokens for DS-1.5B). Given this characteristic, we evaluate three length penalty coefficients ($\beta \in \{5\text{e-}4, 1\text{e-}3, 5\text{e-}3\}$) and report the metrics at training step 150, as detailed in Table 10.

Empirical results demonstrate that APR improves reasoning accuracy while strictly constraining output length compared to the original baseline. Specifically, under the $\beta = 5\text{e-}4$ configuration, the model reduces the overall average response length by 43.4%, while yielding an absolute gain of 11.6% in overall performance.

Furthermore, an ablation across different β values reveals a clear performance-efficiency trade-off: as β increases, the penalty for redundancy intensifies, leading to a strict monotonic increase in the length reduction ratio (from 43.4% to 47.2%). However, excessive length penalization results in a marginal regression in accuracy gains (decreasing from +11.6% to +9.9%). Consequently, the

AE Score, which balances performance and efficiency, peaks at 1.26 under $\beta = 5\text{e-}4$. These findings not only quantify the operational bounds of the penalty coefficient but also confirm the cross-model generalization capability of APR, demonstrating its efficacy in correcting the structural redundancy prevalent across different LRM architectures.